

(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 © 2021 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 © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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

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

HDFGRTR_EL2: Hypervisor Debug Fine-Grained Read Trap Register

HDFGWTR_EL2: Hypervisor Debug Fine-Grained Write Trap Register

HFGITR_EL2: Hypervisor Fine-Grained Instruction Trap Register

HFGRTR_EL2: Hypervisor Fine-Grained Read Trap Register

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_VMCRR_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_AA64PFR0_EL1](#): AArch64 Processor Feature Register 0

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

[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)

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

MFAR_EL3: PA Fault Address Register

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

[PMBIDR_EL1](#): Profiling Buffer ID Register

PMBLIMITR_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

[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

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

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

PMINTENCLR_EL1: Performance Monitors Interrupt Enable Clear register

PMINTENSET_EL1: Performance Monitors Interrupt Enable Set register

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

PMOVSLR_EL0: Performance Monitors Overflow Flag Status Clear Register

PMOVSSSET_EL0: Performance Monitors Overflow Flag Status Set register

PMSCR_EL1: Statistical Profiling Control Register (EL1)

PMSCR_EL2: Statistical Profiling Control Register (EL2)

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

PMSWINC_EL0: Performance Monitors Software Increment register

PMUSERENR_EL0: Performance Monitors User Enable Register

PMXEVCNTR_EL0: Performance Monitors Selected Event Count Register

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

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)

[S3_<op1>_<Cn>_<Cm>_<op2>](#): IMPLEMENTATION DEFINED registers

[SCR_EL3](#): Secure Configuration Register

[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

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

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>

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

TRCSYNCP: 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

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)

3020/09/2021 1412:5740

Copyright © 2010-2021 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

[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 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 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 IGVC: 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>: 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

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

no old file

htmldiff from-

(new)

ALLINT, All Interrupt Mask Bit

The ALLINT characteristics are:

Purpose

Allows access to the all interrupt mask bit.

Configuration

This register is present only when FEAT_NMI is implemented. Otherwise, direct accesses to ALLINT are UNDEFINED.

Attributes

ALLINT is 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																										ALLINT			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 [63:14]

Reserved, RES0.

ALLINT, bit [13]

All interrupt mask.

ALLINT	Meaning
0b0	When SCTL_ELx.NMI is 1 and execution is at ELx, an IRQ or FIQ interrupt with Superpriority that is targeted to ELx is not masked by PSTATE.I or PSTATE.F, respectively, unless both PSTATE.SP and SCTL_ELx.SPINTMASK are 1. An IRQ or FIQ interrupt without Superpriority that is targeted to ELx is masked.
0b1	When SCTL_ELx.NMI is 1 and execution is at ELx, an IRQ or FIQ interrupt that is targeted to ELx is masked by PSTATE.I or PSTATE.F, respectively, regardless of Superpriority.

When executing at EL0 and SCTL_ELx.NMI is 1, if an IRQ or FIQ interrupt targeted to ELx is masked by PSTATE.I or PSTATE.F, the mask applies only to IRQ or FIQ interrupts without Superpriority. IRQ or FIQ interrupts with Superpriority are not masked.

The value of this bit is set to the inverse value in the SCTL_ELx.SPINTMASK field on taking an exception to ELx.

The reset behavior of this field is:

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

Bits [12:0]

Reserved, RES0.

Accessing ALLINT

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

MRS <Xt>, ALLINT

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0011	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    return Zeros(50):PSTATE.ALLINT:Zeros(13);
elsif PSTATE.EL == EL2 then
    return Zeros(50):PSTATE.ALLINT:Zeros(13);
elsif PSTATE.EL == EL3 then
    return Zeros(50):PSTATE.ALLINT:Zeros(13);
```

MSR ALLINT, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0011	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && IsHCRXEL2Enabled() && HCRX_EL2.TALLINT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PSTATE.ALLINT = X[t]<13>;
elsif PSTATE.EL == EL2 then
    PSTATE.ALLINT = X[t]<13>;
elsif PSTATE.EL == EL3 then
    PSTATE.ALLINT = X[t]<13>;
```

MSR ALLINT, #<imm>

op0	op1	CRn	op2
0b00	0b001	0b0100	0b000

30/09/2021 14:52; 092b4e1bbfbb45a293b198f9330c5f529ead2b0f

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AT S12E0R, Address Translate Stages 1 and 2 EL0 Read

The AT S12E0R characteristics are:

Purpose

Performs stage 1 and 2 address translations 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 S12E0R 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 S12E0R instruction

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

AT S12E0R, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1000	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
    if HCR_EL2.<E2H,TGE> == '11' || HCR_EL2.<DC,VM> == '00' then
        AArch64.ATAT_S1E0R(X[t], TranslationStage_1, EL0, ATAccess_Read);});
    else
        AArch64.ATAT_S12E0R(X[t], TranslationStage_12, EL0, ATAccess_Read);});
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        AArch64.ATAT_S1E0R(X[t], TranslationStage_1, EL0, ATAccess_Read);});
    elseif EL2Enabled() && (HCR_EL2.<E2H,TGE> == '11' || HCR_EL2.<DC,VM> == '00') then
        AArch64.ATAT_S1E0R(X[t], TranslationStage_1, EL0, ATAccess_Read);});
    else
        AArch64.ATAT_S12E0R(X[t], TranslationStage_12, EL0, ATAccess_Read);});

```

3020/09/2021 14:53:37: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AT S12E0W, Address Translate Stages 1 and 2 EL0 Write

The AT S12E0W characteristics are:

Purpose

Performs stage 1 and 2 address translations 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 S12E0W 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 S12E0W instruction

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

AT S12E0W, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1000	0b111


```

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
    if HCR_EL2.<E2H,TGE> == '11' || HCR_EL2.<DC,VM> == '00' then
        AArch64.ATAT_S1E0W(X[t], TranslationStage_1, EL0, ATAccess_Write);});
    else
        AArch64.ATAT_S12E0W(X[t], TranslationStage_12, EL0, ATAccess_Write);});
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        AArch64.ATAT_S1E0W(X[t], TranslationStage_1, EL0, ATAccess_Write);});
    elseif EL2Enabled() && (HCR_EL2.<E2H,TGE> == '11' || HCR_EL2.<DC,VM> == '00') then
        AArch64.ATAT_S1E0W(X[t], TranslationStage_1, EL0, ATAccess_Write);});
    else
        AArch64.ATAT_S12E0W(X[t], TranslationStage_12, EL0, ATAccess_Write);});

```

3020/09/2021 14:52:37: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AT S12E1R, Address Translate Stages 1 and 2 EL1 Read

The AT S12E1R characteristics are:

Purpose

Performs stage 1 and 2 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 S12E1R 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 S12E1R instruction

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

AT S12E1R, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1000	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 HCR_EL2.<E2H,TGE> == '11' || HCR_EL2.<DC,VM> == '00' then
        AArch64.ATAT_S1E1R(X[t], TranslationStage_1, EL1, ATAccess_Read);});
    else
        AArch64.ATAT_S12E1R(X[t], TranslationStage_12, EL1, ATAccess_Read);});
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        AArch64.ATAT_S1E1R(X[t], TranslationStage_1, EL1, ATAccess_Read);});
    elsif EL2Enabled() && (HCR_EL2.<E2H,TGE> == '11' || HCR_EL2.<DC,VM> == '00') then
        AArch64.ATAT_S1E1R(X[t], TranslationStage_1, EL1, ATAccess_Read);});
    else
        AArch64.ATAT_S12E1R(X[t], TranslationStage_12, EL1, ATAccess_Read);});

```

3020/09/2021 14:12:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AT S12E1W, Address Translate Stages 1 and 2 EL1 Write

The AT S12E1W characteristics are:

Purpose

Performs stage 1 and 2 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 S12E1W 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 S12E1W instruction

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

AT S12E1W, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1000	0b101

```

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
    if HCR_EL2.<E2H,TGE> == '11' || HCR_EL2.<DC,VM> == '00' then
        AArch64.ATAT_S1E1W(X[t], TranslationStage_1, EL1, ATAccess_Write);});
    else
        AArch64.ATAT_S12E1W(X[t], TranslationStage_12, EL1, ATAccess_Write);});
elseif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        AArch64.ATAT_S1E1W(X[t], TranslationStage_1, EL1, ATAccess_Write);});
    elseif EL2Enabled() && (HCR_EL2.<E2H,TGE> == '11' || HCR_EL2.<DC,VM> == '00') then
        AArch64.ATAT_S1E1W(X[t], TranslationStage_1, EL1, ATAccess_Write);});
    else
        AArch64.ATAT_S12E1W(X[t], TranslationStage_12, EL1, ATAccess_Write);});

```

3020/09/2021 14:53:37: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.ATS1E0R == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.ATAT_S1E0R(X[t], TranslationStage_1, EL0, ATAccess_Read);
elsif PSTATE.EL == EL2 then
    AArch64.ATAT_S1E0R(X[t], TranslationStage_1, EL0, ATAccess_Read);
elsif PSTATE.EL == EL3 then
    AArch64.ATAT_S1E0R(X[t], TranslationStage_1, EL0, ATAccess_Read);

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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.

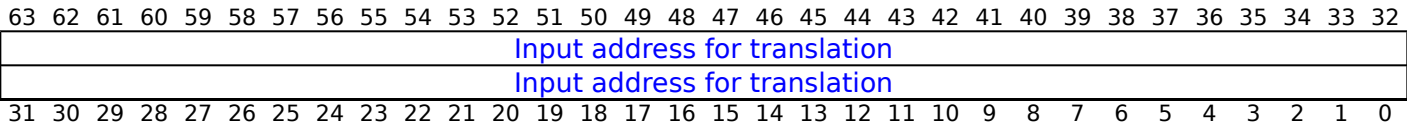
Configuration

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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.ATS1E0W == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.ATAT_S1E0W(X[t], TranslationStage_1, EL0, ATAccess_Write);});
elsif PSTATE.EL == EL2 then
    AArch64.ATAT_S1E0W(X[t], TranslationStage_1, EL0, ATAccess_Write);});
elsif PSTATE.EL == EL3 then
    AArch64.ATAT_S1E0W(X[t], TranslationStage_1, EL0, ATAccess_Write);});

```

3020/09/2021 1412:5236; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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																															
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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.ATS1E1R == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.ATAT_S1E1R(X[t], TranslationStage_1, EL1, ATAccess_Read);});
elsif PSTATE.EL == EL2 then
    AArch64.ATAT_S1E1R(X[t], TranslationStage_1, EL1, ATAccess_Read);});
elsif PSTATE.EL == EL3 then
    AArch64.ATAT_S1E1R(X[t], TranslationStage_1, EL1, ATAccess_Read);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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.

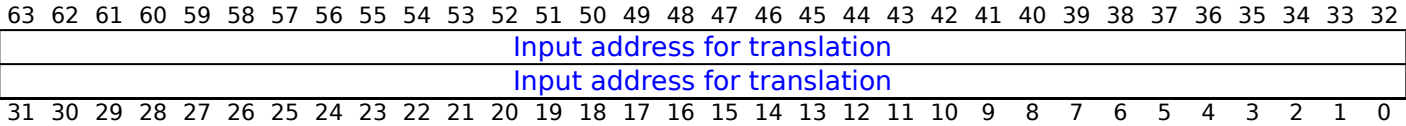
Configuration

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



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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.ATS1E1RP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.ATAT_S1E1RP(X[t], TranslationStage_1, EL1, ATAccess_ReadPAN);});
elsif PSTATE.EL == EL2 then
    AArch64.ATAT_S1E1RP(X[t], TranslationStage_1, EL1, ATAccess_ReadPAN);});
elsif PSTATE.EL == EL3 then
    AArch64.ATAT_S1E1RP(X[t], TranslationStage_1, EL1, ATAccess_ReadPAN);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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

63	62	61	60	59	58	57	56	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 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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.ATS1E1W == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.ATAT_S1E1W(X[t], TranslationStage_1, EL1, ATAccess_Write);});
elsif PSTATE.EL == EL2 then
    AArch64.ATAT_S1E1W(X[t], TranslationStage_1, EL1, ATAccess_Write);});
elsif PSTATE.EL == EL3 then
    AArch64.ATAT_S1E1W(X[t], TranslationStage_1, EL1, ATAccess_Write);});

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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.

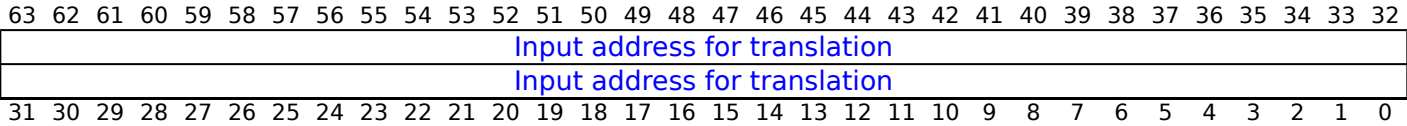
Configuration

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



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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.ATS1E1WP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.ATAT_S1E1WP(X[t], TranslationStage_1, EL1, ATAccess_WritePAN);});
elsif PSTATE.EL == EL2 then
    AArch64.ATAT_S1E1WP(X[t], TranslationStage_1, EL1, ATAccess_WritePAN);});
elsif PSTATE.EL == EL3 then
    AArch64.ATAT_S1E1WP(X[t], TranslationStage_1, EL1, ATAccess_WritePAN);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AT S1E2R, Address Translate Stage 1 EL2 Read

The AT S1E2R characteristics are:

Purpose

Performs stage 1 address translation as defined for EL2, with permissions as if reading from the given virtual address.

Configuration

There are no configuration notes.

Attributes

AT S1E2R 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 S1E2R instruction

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

AT S1E2R, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1000	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.ATAT_S1E2R(X[t], TranslationStage_1, EL2, ATAccess_Read);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    else
        AArch64.ATAT_S1E2R(X[t], TranslationStage_1, EL2, ATAccess_Read);

```

3020/09/2021 1412:5237: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

(old)

htmldiff from-

(new)

AT S1E2W, Address Translate Stage 1 EL2 Write

The AT S1E2W characteristics are:

Purpose

Performs stage 1 address translation as defined for EL2, with permissions as if writing to the given virtual address.

Configuration

There are no configuration notes.

Attributes

AT S1E2W 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 S1E2W instruction

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

AT S1E2W, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b100	0b0111	0b1000	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
    AArch64.ATAT_S1E2W(X[t], TranslationStage_1, EL2, ATAccess_Write);
elsif PSTATE.EL == EL3 then
    if !EL2Enabled() then
        UNDEFINED;
    else
        AArch64.ATAT_S1E2W(X[t], TranslationStage_1, EL2, ATAccess_Write);

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

AT S1E3R, Address Translate Stage 1 EL3 Read

The AT S1E3R characteristics are:

Purpose

Performs stage 1 address translation as defined for EL3, with permissions as if reading from the given virtual address.

Configuration

There are no configuration notes.

Attributes

AT S1E3R 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 S1E3R instruction

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

AT S1E3R, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b110	0b0111	0b1000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    AArch64.ATAT_S1E3R(X[t], TranslationStage_1, EL3, ATAccess_Read);

```

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

(old)

htmldiff from-

(new)

AT S1E3W, Address Translate Stage 1 EL3 Write

The AT S1E3W characteristics are:

Purpose

Performs stage 1 address translation as defined for EL3, with permissions as if writing to the given virtual address.

Configuration

There are no configuration notes.

Attributes

AT S1E3W 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
Input address for translation																															

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 S1E3W instruction

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

AT S1E3W, <Xt>

op0	op1	CRn	CRm	op2
0b01	0b110	0b0111	0b1000	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.ATAT_S1E3W(X[t], TranslationStage_1, EL3, ATAccess_Write);
end if

```


(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

BRBCR_EL1, Branch Record Buffer Control Register (EL1)

The BRBCR_EL1 characteristics are:

Purpose

Controls the Branch Record Buffer.

Configuration

This register is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRBCR_EL1 are UNDEFINED.

Attributes

BRBCR_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				EXCEPTION				ERTN				RES0								FZP		RES0		TS		MPRED		CC		RES0		E1BRE		EOBRE	
31	30	29	28	27	26	25	24	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.

EXCEPTION, bit [23]

Enable the recording of entry to EL1 via an exception.

EXCEPTION	Meaning
0b0	Disable the recording of Branch records for exceptions when taken to EL1.
0b1	Enable the recording of Branch records for exceptions when taken to EL1.

The reset behavior of this field is:

- On a ColdWarm 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.

ERTN, bit [22]

Allow the recording Branch records for exception return instructions from EL1.

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~~Warm 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 [21:9]

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~~Warm 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() && (!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
            return NVMem[0x8E0];
        else
            return 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
            return BRBCR_EL2;
        else
            return BRBCR_EL1;
    elsif PSTATE.EL == EL3 then
        return BRBCR_EL1;

```

MRS <Xt>, BRBCR_EL12

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
        return NVMem[0x8E0];
    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
            return BRBCR_EL1;
    else
        UNDEFINED;
elseif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        return 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() && (!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];
    else
        BRBCR_EL1 = X[t];
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];
    else
        BRBCR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    BRBCR_EL1 = X[t];

```

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];
    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];
    else
        UNDEFINED;
elseif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        BRBCR_EL1 = X[t];
    else
        UNDEFINED;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

BRBCR_EL2, Branch Record Buffer Control Register (EL2)

The BRBCR_EL2 characteristics are:

Purpose

Controls the Branch Record Buffer.

Configuration

This register is present only when FEAT_BRBE is implemented. Otherwise, direct accesses to BRBCR_EL2 are UNDEFINED.

Attributes

BRBCR_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				EXCEPTION				ERTN				RES0								FZP		RES0		TS		MPRED		CC		RES0		E2BRE		E0HBRE	
31	30	29	28	27	26	25	24	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.

EXCEPTION, bit [23]

Enable the recording of entry to EL2 via an exception.

EXCEPTION	Meaning
0b0	Disable the recording of Branch records for exceptions when taken to EL2.
0b1	Enable the recording of Branch records for exceptions when taken to EL2.

The reset behavior of this field is:

- On a ColdWarm 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.

ERTN, bit [22]

Allow the recording Branch records for exception return instructions from EL2.

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 [21:9]

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.

EOHBRE, bit [0]

EL0 Branch recording enable.

EOHBRE	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() && (!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
        return NVMem[0x8E0];
    else
        return 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
        return BRBCR_EL2;
    else
        return BRBCR_EL1;
elsif PSTATE.EL == EL3 then
    return 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
        return BRBCR_EL2;
elsif PSTATE.EL == EL3 then
    return 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() && (!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];
    else
        BRBCR_EL1 = X[t];
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];
    else
        BRBCR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    BRBCR_EL1 = X[t];

```

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];
elsif PSTATE.EL == EL3 then
    BRBCR_EL2 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The BRBF_{CR_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 BRBFCR_EL1 are UNDEFINED.

Attributes

BRBFCR_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
RES0																											
RES0	BANK	RES0		CONDDIR	DIRCALL	INDCALL	RTN	INDIRECT	DIRECT	Eni	RES0				PAUSED	LAST		FAILED	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

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 **ColdWarm** 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 **ColdWarm** 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 **ColdWarm** 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 **ColdWarm** 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 **ColdWarm** 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 ColdWarm 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 ColdWarm 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 ColdWarm 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~~ Warm 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() && (!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
        return BRBFCCR_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
        return BRBFCCR_EL1;
elsif PSTATE.EL == EL3 then
    return BRBFCCR_EL1;

```

MSR BRBFCCR_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() && (!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];
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];
elsif PSTATE.EL == EL3 then
    BRBFCR_EL1 = X[t];

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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:

- **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:

- **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.	
0b01	EL1.	
0b10	EL2.	
0b11	EL3.	When FEAT_BRBEv1p1 is implemented

All other values are reserved.

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:

- 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:

- 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.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() && (!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
        return 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
        return BRBINFINJ_EL1;
elsif PSTATE.EL == EL3 then
    return BRBINFINJ_EL1;

```

MSR BRBINFINJ_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b1001	0b0001	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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_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
        BRBINFINJ_EL1 = X[t];
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
        BRBINFINJ_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    BRBINFINJ_EL1 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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 + (BRBFECR_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:

- **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:

- 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

All other values are reserved.

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:

- 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:

- 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.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<n>_EL1

op0	op1	CRn	CRm	op2
0b10	0b001	0b1000	n[3:0]	n[4]:0b00

```

if PSTATE.EL == 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() && (!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
                return BRBINF_EL1[UInt(op2<2>:CRm<3:0>)];
        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
                        return BRBINF_EL1[UInt(op2<2>:CRm<3:0>)];
        elsif PSTATE.EL == EL3 then
            return BRBINF_EL1[UInt(op2<2>:CRm<3:0>)];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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	Secure.
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}.

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}.

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 ASID 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' && (!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.RestrictPredictionCFP_RCTX(X[t], RestrictType_ControlFlow);});
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.NV == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.CFPRCTX == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.RestrictPredictionCFP_RCTX(X[t], RestrictType_ControlFlow);});
    elsif PSTATE.EL == EL2 then
        AArch64.RestrictPredictionCFP_RCTX(X[t], RestrictType_ControlFlow);});
    elsif PSTATE.EL == EL3 then
        AArch64.RestrictPredictionCFP_RCTX(X[t], RestrictType_ControlFlow);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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.

The possible values are:

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.

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.

LoUIS, bits [23:21]

Level of Unification Inner Shareable for the cache hierarchy.

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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.CLIDR_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CLIDR_EL1;
    elsif PSTATE.EL == EL2 then
        return CLIDR_EL1;
    elsif PSTATE.EL == EL3 then
        return CLIDR_EL1;

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10
RES0												CNTPMASK	CNTVMASK	EVNTIS	EL1NVVCT	EL1NVPCT	EL1TVCT	EL1TVT	ECV	EL1PTEN	EL1PCTE

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 . CNTV_CTL_EL0.IMASK behaves as if set to 1 for all purposes other than a direct read of the field.
0b1	

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	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: <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.ELOVTEN. 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.ELOVTEN or CNTKCTL.PLOVTEN.

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<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 EL1 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.PL0PTEN .
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 ~~the counter register~~ [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 ~~the counter register~~ [CNTPCT_EL0](#).

Otherwise, this field selects a trigger bit in the range 0 to 15 of ~~the counter register~~ [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 ~~counter register~~ [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 ~~the counter register~~ [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.

EL0VCTEN	Meaning
0b0	<p>EL0 using AArch64: EL0 accesses to the CNTVCT_EL0 are trapped to EL2.</p> <p>EL0 using AArch64: EL0 accesses to the CNTFRQ_EL0 register are trapped to EL2, if CNTHCTL_EL2.EL0PCTEN is also 0.</p> <p>EL0 using AArch32: EL0 accesses to the CNTVCT are trapped to EL2.</p> <p>EL0 using AArch32: EL0 accesses to the CNTFRQ register are trapped to EL2, if CNTHCTL.EL0PCTEN is also 0.</p>
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.

ELOPCTEN, 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	<p>EL0 using AArch64: EL0 accesses to the CNTPCT_EL0 are trapped to EL2.</p> <p>EL0 using AArch64: EL0 accesses to the CNTFRQ_EL0 register are trapped to EL2, if CNTHCTL_EL2.EL0VCTEN is also 0.</p> <p>EL0 using AArch32: EL0 accesses to the CNTPCT are trapped to EL2.</p> <p>EL0 using AArch32: EL0 accesses to the CNTFRQ and register are trapped to EL2, if CNTHCTL_EL2.EL0VCTEN is also 0.</p>
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
																RES0															
RES0																CNTPMASK	CNTVMASK	EVNTIS	EL1INVCT	EL1INVPCT	EL1TVCT	EL1TVT	ECV	RES0	EVNTI						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4				

This format applies in all Armv8.0 implementations, and it also contains a description of 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_ELO .IMASK.
0b1	CNTP_CTL_ELO .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_ELO .IMASK.
0b1	CNTV_CTL_ELO .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_ELO [15:0].
0b1	The CNTHCTL_EL2.EVENTI 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	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: <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 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 ~~the counter register~~ [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 ~~the counter register~~ [CNTPCT_EL0](#).

Otherwise, this field selects a trigger bit in the range 0 to 15 of ~~the counter register~~ [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 ~~counter register~~ [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 ~~the counter register~~ [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
    return CNTHCTL_EL2;
elsif PSTATE.EL == EL3 then
    return 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];
elsif PSTATE.EL == EL3 then
    CNTHCTL_EL2 = X[t];

```

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
    return CNTKCTL_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return CNTHCTL_EL2;
    else
        return CNTKCTL_EL1;
elsif PSTATE.EL == EL3 then
    return 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];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        CNTHCTL_EL2 = X[t];
    else
        CNTKCTL_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    CNTKCTL_EL1 = X[t];

```

30/09/2021 14:53; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The CNTHP_CTL_EL2 characteristics are:

Purpose

Control register for the EL2 physical timer.

Configuration

AArch64 System register CNTHP_CTL_EL2 bits [31:0] are architecturally mapped to AArch32 System register [CNTHP_CTL\[31:0\]](#).

This register is present only when EL3 is implemented or (EL3 is not implemented, EL2 is implemented and FEAT_SEL2 is not implemented). Otherwise, direct accesses to CNTHP_CTL_EL2 are UNDEFINED.

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

Attributes

CNTHP_CTL_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					ISTATUS		IMASK		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 [63:3]

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 [CNTHP_TVAL_EL2](#) 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 CNTHP_CTL_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic CNTHP_CTL_EL2 or CNTP_CTL_EL0 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>, CNTHP_CTL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	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
    return CNTHP_CTL_EL2;
elsif PSTATE.EL == EL3 then
    return CNTHP_CTL_EL2;

```

MSR CNTHP_CTL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	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
    CNTHP_CTL_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    CNTHP_CTL_EL2 = X[t];

```

MRS <Xt>, CNTP_CTL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
        IsFeatureImplemented(FEAT_SEL2) then
        return CNTHPS_CTL_EL2;
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        return CNTHP_CTL_EL2;
    else
        return CNTP_CTL_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x180];
    else
        return CNTP_CTL_EL0;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        return CNTHPS_CTL_EL2;
    elsif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        return CNTHP_CTL_EL2;
    else
        return CNTP_CTL_EL0;
elsif PSTATE.EL == EL3 then
    return CNTP_CTL_EL0;

```

MSR CNTP_CTL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHPS_CTL_EL2 = X[t];
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        CNTHP_CTL_EL2 = X[t];
    else
        CNTP_CTL_EL0 = X[t];
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x180] = X[t];
    else
        CNTP_CTL_EL0 = X[t];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        CNTHPS_CTL_EL2 = X[t];
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        CNTHP_CTL_EL2 = X[t];
    else
        CNTP_CTL_EL0 = X[t];
elseif PSTATE.EL == EL3 then
    CNTP_CTL_EL0 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbfdb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTHP_CVAL_EL2, Counter-timer Physical Timer CompareValue register (EL2)

The CNTHP_CVAL_EL2 characteristics are:

Purpose

Holds the compare value for the EL2 physical timer.

Configuration

AArch64 System register CNTHP_CVAL_EL2 bits [63:0] are architecturally mapped to AArch32 System register [CNTHP_CVAL\[63:0\]](#).

This register is present only when EL3 is implemented or (EL3 is not implemented, EL2 is implemented and FEAT_SEL2 is not implemented). Otherwise, direct accesses to CNTHP_CVAL_EL2 are UNDEFINED.

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

Attributes

CNTHP_CVAL_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
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																															

CompareValue, bits [63:0]

Holds the EL2 physical timer CompareValue.

When [CNTHP_CTL_EL2](#).ENABLE is 1, the timer condition is met when ([CNTPTCT_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:

- [CNTHP_CTL_EL2](#).ISTATUS is set to 1.
- If [CNTHP_CTL_EL2](#).IMASK is 0, an interrupt is generated.

When [CNTHP_CTL_EL2](#).ENABLE is 0, the timer condition is not met, but [CNTPTCT_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 CNTHP_CVAL_EL2

When [HCR_EL2](#).E2H is 1, without explicit synchronization, access from EL2 using the mnemonic CNTHP_CVAL_EL2 or CNTP_CVAL_EL0 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>, CNTHP_CVAL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0010	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
    return CNTHP_CVAL_EL2;
elsif PSTATE.EL == EL3 then
    return CNTHP_CVAL_EL2;

```

MSR CNTHP_CVAL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0010	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
    CNTHP_CVAL_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    CNTHP_CVAL_EL2 = X[t];

```

MRS <Xt>, CNTP_CVAL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b010

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        return CNTHPS_CVAL_EL2;
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        return CNTHP_CVAL_EL2;
    else
        return CNTP_CVAL_EL0;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x178];
    else
        return CNTP_CVAL_EL0;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        return CNTHPS_CVAL_EL2;
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        return CNTHP_CVAL_EL2;
    else
        return CNTP_CVAL_EL0;
elseif PSTATE.EL == EL3 then
    return CNTP_CVAL_EL0;

```

MSR CNTP_CVAL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b010

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHPS_CVAL_EL2 = X[t];
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        CNTHP_CVAL_EL2 = X[t];
    else
        CNTP_CVAL_EL0 = X[t];
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x178] = X[t];
    else
        CNTP_CVAL_EL0 = X[t];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        CNTHPS_CVAL_EL2 = X[t];
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        CNTHP_CVAL_EL2 = X[t];
    else
        CNTP_CVAL_EL0 = X[t];
elseif PSTATE.EL == EL3 then
    CNTP_CVAL_EL0 = X[t];

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd536e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTHP_TVAL_EL2, Counter-timer Physical Timer TimerValue register (EL2)

The CNTHP_TVAL_EL2 characteristics are:

Purpose

Holds the timer value for the EL2 physical timer.

Configuration

AArch64 System register CNTHP_TVAL_EL2 bits [31:0] are architecturally mapped to AArch32 System register [CNTHP_TVAL\[31:0\]](#).

This register is present only when EL3 is implemented or (EL3 is not implemented, EL2 is implemented and FEAT_SEL2 is not implemented). Otherwise, direct accesses to CNTHP_TVAL_EL2 are UNDEFINED.

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

Attributes

CNTHP_TVAL_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																															
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 EL2 physical timer.

On a read of this register:

- If [CNTHP_CTL_EL2.ENABLE](#) is 0, the value returned is UNKNOWN.
- If [CNTHP_CTL_EL2.ENABLE](#) is 1, the value returned is ([CNTHP_CVAL_EL2](#) - [CNTPCT_EL0](#)).

On a write of this register, [CNTHP_CVAL_EL2](#) is set to ([CNTPCT_EL0](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTHP_CTL_EL2.ENABLE](#) is 1, the timer condition is met when ([CNTPCT_EL0](#) - [CNTHP_CVAL_EL2](#)) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTHP_CTL_EL2.ISTATUS](#) is set to 1.
- If [CNTHP_CTL_EL2.IMASK](#) is 0, an interrupt is generated.

When [CNTHP_CTL_EL2.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 CNTHP_TVAL_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic CNTHP_TVAL_EL2 or CNTP_TVAL_EL0 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>, CNTHP_TVAL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	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 CNTHP_CTL_EL2.ENABLE == '0' then
        return bits(64) UNKNOWNCNTHP_TVAL_EL2;
    else
        return CNTHP_CVAL_EL2 - PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    if CNTHP_CTL_EL2.ENABLE == '0' then
        return bits(64) UNKNOWNCNTHP_TVAL_EL2;
    else
        return CNTHP_CVAL_EL2 - PhysicalCountInt();

```

MSR CNTHP_TVAL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	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
    CNTHP_CVAL_EL2CNTHP_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    CNTHP_CVAL_EL2CNTHP_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();

```

MRS <Xt>, CNTP_TVAL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b000

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHPS_TVAL_EL2;
        else
            return CNTHPS_CVAL_EL2 - PhysicalCountInt();
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        if CNTHP_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHP_TVAL_EL2;
        else
            return CNTHP_CVAL_EL2 - PhysicalCountInt();
    elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11' then
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWN;
        else
            return CNTP_CVAL_EL0 - (PhysicalCountInt() - CNTPOFF_EL2);
    else
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTP_TVAL_EL0;
        else
            return CNTP_CVAL_EL0 - PhysicalCountInt();
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' then
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWN;
        else
            return CNTP_CVAL_EL0 - (PhysicalCountInt() - CNTPOFF_EL2);
    else
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTP_TVAL_EL0;
        else
            return CNTP_CVAL_EL0 - PhysicalCountInt();
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHPS_TVAL_EL2;
        else
            return CNTHPS_CVAL_EL2 - PhysicalCountInt();
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        if CNTHP_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHP_TVAL_EL2;
        else
            return CNTHP_CVAL_EL2 - PhysicalCountInt();
    else
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTP_TVAL_EL0;
        else
            return CNTP_CVAL_EL0 - PhysicalCountInt();
elseif PSTATE.EL == EL3 then
    if CNTP_CTL_EL0.ENABLE == '0' then
        return bits(64) UNKNOWNCNTP_TVAL_EL0;
    else

```

```
return CNTP_CVAL_EL0 - PhysicalCountInt();
```

MSR CNTP_TVAL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b000

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elseif EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
        IsFeatureImplemented(FEAT_SEL2) then
            CNTHPS_CVAL_EL2CNTHPS_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
        elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
            CNTHP_CVAL_EL2CNTHP_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
        elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
        CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11' then
            CNTP_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTPOFF_EL2;
        else
            CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif EL2Enabled() && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
        CNTHCTL_EL2.ECV == '1' then
            CNTP_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTPOFF_EL2;
        else
            CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
    elseif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
            CNTHPS_CVAL_EL2CNTHPS_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
        elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
            CNTHP_CVAL_EL2CNTHP_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
        else
            CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
    elseif PSTATE.EL == EL3 then
        CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd36e47856e443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTHPS_TVAL_EL2, Counter-timer Secure Physical Timer TimerValue register (EL2)

The CNTHPS_TVAL_EL2 characteristics are:

Purpose

Holds the timer value for the Secure EL2 physical timer.

Configuration

AArch64 System register CNTHPS_TVAL_EL2 bits [31:0] are architecturally mapped to AArch32 System register [CNTHPS_TVAL\[31:0\]](#).

This register is present only when EL2 is implemented and FEAT_SEL2 is implemented. Otherwise, direct accesses to CNTHPS_TVAL_EL2 are UNDEFINED.

Attributes

CNTHPS_TVAL_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																															
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 EL2 physical timer.

On a read of this register:

- If [CNTHPS_CTL_EL2](#).ENABLE is 0, the value returned is UNKNOWN.
- If [CNTHPS_CTL_EL2](#).ENABLE is 1, the value returned is ([CNTHPS_CVAL_EL2](#) - [CNTPTCT_EL0](#)).

On a write of this register, [CNTHPS_CVAL_EL2](#) is set to ([CNTPTCT_EL0](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTHPS_CTL_EL2](#).ENABLE is 1, the timer condition is met when ([CNTPTCT_EL0](#) - [CNTHPS_CVAL_EL2](#)) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTHPS_CTL_EL2](#).ISTATUS is set to 1.
- If [CNTHPS_CTL_EL2](#).IMASK is 0, an interrupt is generated.

When [CNTHPS_CTL_EL2](#).ENABLE is 0, the timer condition is not met, but [CNTPTCT_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 CNTHPS_TVAL_EL2

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

MRS <Xt>, CNTHPS_TVAL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0101	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsSecure() then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsSecure() then
        UNDEFINED;
    else
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWN_CNTHPS_TVAL_EL2;
        else
            return CNTHPS_CVAL_EL2 - PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWN_CNTHPS_TVAL_EL2;
        else
            return CNTHPS_CVAL_EL2 - PhysicalCountInt();

```

MSR CNTHPS_TVAL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0101	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsSecure() then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsSecure() then
        UNDEFINED;
    else
        CNTHPS_CVAL_EL2_CNTHPS_TVAL_EL2 = SignExtend((X[t]<31:0>), 64) + PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        CNTHPS_CVAL_EL2_CNTHPS_TVAL_EL2 = SignExtend((X[t]<31:0>), 64) + PhysicalCountInt();

```

MRS <Xt>, CNTP_TVAL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b000

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHPS_TVAL_EL2;
        else
            return CNTHPS_CVAL_EL2 - PhysicalCountInt();
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        if CNTHP_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHP_TVAL_EL2;
        else
            return CNTHP_CVAL_EL2 - PhysicalCountInt();
    elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11' then
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWN;
        else
            return CNTP_CVAL_EL0 - (PhysicalCountInt() - CNTPOFF_EL2);
    else
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTP_TVAL_EL0;
        else
            return CNTP_CVAL_EL0 - PhysicalCountInt();
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' then
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWN;
        else
            return CNTP_CVAL_EL0 - (PhysicalCountInt() - CNTPOFF_EL2);
    else
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTP_TVAL_EL0;
        else
            return CNTP_CVAL_EL0 - PhysicalCountInt();
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHPS_TVAL_EL2;
        else
            return CNTHPS_CVAL_EL2 - PhysicalCountInt();
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        if CNTHP_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHP_TVAL_EL2;
        else
            return CNTHP_CVAL_EL2 - PhysicalCountInt();
    else
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTP_TVAL_EL0;
        else
            return CNTP_CVAL_EL0 - PhysicalCountInt();
elseif PSTATE.EL == EL3 then
    if CNTP_CTL_EL0.ENABLE == '0' then
        return bits(64) UNKNOWNCNTP_TVAL_EL0;
    else
        return CNTP_CVAL_EL0 - PhysicalCountInt();

```

```
return CNTP_CVAL_EL0 - PhysicalCountInt();
```

MSR CNTP_TVAL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b000

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elseif EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
            CNTHPS_CVAL_EL2CNTHPS_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();;;
        elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
            CNTHP_CVAL_EL2CNTHP_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();;;
        elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11' then
            CNTP_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTPOFF_EL2;
        else
            CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();;;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif EL2Enabled() && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' then
            CNTP_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTPOFF_EL2;
        else
            CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();;;
    elseif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
            CNTHPS_CVAL_EL2CNTHPS_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();;;
        elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
            CNTHP_CVAL_EL2CNTHP_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();;;
        else
            CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();;;
    elseif PSTATE.EL == EL3 then
        CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();;;
```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd9b36e47856e443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The CNTHV_CTL_EL2 characteristics are:

Purpose

Control register for the EL2 virtual timer.

Configuration

AArch64 System register CNTHV_CTL_EL2 bits [31:0] are architecturally mapped to AArch32 System register [CNTHV_CTL\[31:0\]](#).

This register is present only when FEAT_VHE is implemented and (EL3 is implemented or (EL3 is not implemented and FEAT_SEL2 is not implemented)). Otherwise, direct accesses to CNTHV CTL EL2 are UNDEFINED.

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

Attributes

CNTHV_CTL_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																										ISTATUS		IMASK		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 [63:3]

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 [CNTHV_TVAL_EL2](#) 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 CNTHV_CTL_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic CNTHV_CTL_EL2 or CNTV_CTL_EL0 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>, CNTHV_CTL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	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
    return CNTHV_CTL_EL2;
elsif PSTATE.EL == EL3 then
    return CNTHV_CTL_EL2;

```

MSR CNTHV_CTL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	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
    CNTHV_CTL_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    CNTHV_CTL_EL2 = X[t];

```

MRS <Xt>, CNTV_CTL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
        IsFeatureImplemented(FEAT_SEL2) then
        return CNTHVS_CTL_EL2;
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        return CNTHV_CTL_EL2;
    else
        return CNTV_CTL_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x170];
    else
        return CNTV_CTL_EL0;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        return CNTHVS_CTL_EL2;
    elsif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        return CNTHV_CTL_EL2;
    else
        return CNTV_CTL_EL0;
elsif PSTATE.EL == EL3 then
    return CNTV_CTL_EL0;

```

MSR CNTV_CTL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CTL_EL2 = X[t];
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        CNTHV_CTL_EL2 = X[t];
    else
        CNTV_CTL_EL0 = X[t];
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x170] = X[t];
    else
        CNTV_CTL_EL0 = X[t];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CTL_EL2 = X[t];
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        CNTHV_CTL_EL2 = X[t];
    else
        CNTV_CTL_EL0 = X[t];
elseif PSTATE.EL == EL3 then
    CNTV_CTL_EL0 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdff36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

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

The CNTHV_CVAL_EL2 characteristics are:

Purpose

Holds the compare value for the EL2 virtual timer.

Configuration

AArch64 System register CNTHV_CVAL_EL2 bits [63:0] are architecturally mapped to AArch32 System register [CNTHV_CVAL\[63:0\]](#).

This register is present only when FEAT_VHE is **implemented and (EL3 is implemented or (EL3 is not implemented and FEAT_SEL2 is not implemented)).implemented**. Otherwise, direct accesses to CNTHV_CVAL_EL2 are UNDEFINED.

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

Attributes

CNTHV_CVAL_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
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																															

CompareValue, bits [63:0]

Holds the EL2 virtual timer CompareValue.

When [CNTHV_CTL_EL2](#).ENABLE is 1, the timer condition is met when ([CNTVCT_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:

- [CNTHV_CTL_EL2](#).ISTATUS is set to 1.
- If [CNTHV_CTL_EL2](#).IMASK is 0, an interrupt is generated.

When [CNTHV_CTL_EL2](#).ENABLE is 0, the timer condition is not met, but [CNTVCT_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 CNTHV_CVAL_EL2

When [HCR_EL2](#).E2H is 1, without explicit synchronization, access from EL2 using the mnemonic CNTHV_CVAL_EL2 or CNTV_CVAL_EL0 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>, CNTHV_CVAL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0011	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
    return CNTHV_CVAL_EL2;
elsif PSTATE.EL == EL3 then
    return CNTHV_CVAL_EL2;

```

MSR CNTHV_CVAL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0011	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
    CNTHV_CVAL_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    CNTHV_CVAL_EL2 = X[t];

```

MRS <Xt>, CNTV_CVAL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b010

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        return CNTHVS_CVAL_EL2;
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        return CNTHV_CVAL_EL2;
    else
        return CNTV_CVAL_EL0;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x168];
    else
        return CNTV_CVAL_EL0;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        return CNTHVS_CVAL_EL2;
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        return CNTHV_CVAL_EL2;
    else
        return CNTV_CVAL_EL0;
elseif PSTATE.EL == EL3 then
    return CNTV_CVAL_EL0;

```

MSR CNTV_CVAL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b010

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2 = X[t];
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        CNTHV_CVAL_EL2 = X[t];
    else
        CNTV_CVAL_EL0 = X[t];
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x168] = X[t];
    else
        CNTV_CVAL_EL0 = X[t];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2 = X[t];
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        CNTHV_CVAL_EL2 = X[t];
    else
        CNTV_CVAL_EL0 = X[t];
elseif PSTATE.EL == EL3 then
    CNTV_CVAL_EL0 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

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

The CNTHV_TVAL_EL2 characteristics are:

Purpose

Holds the timer value for the EL2 virtual timer.

Configuration

AArch64 System register CNTHV_TVAL_EL2 bits [31:0] are architecturally mapped to AArch32 System register [CNTHV_TVAL\[31:0\]](#).

This register is present only when FEAT_VHE is **implemented and (EL3 is implemented or (EL3 is not implemented and FEAT_SEL2 is not implemented)).implemented**. Otherwise, direct accesses to CNTHV_TVAL_EL2 are UNDEFINED.

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

Attributes

CNTHV_TVAL_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																															
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 EL2 virtual timer.

On a read of this register:

- If [CNTHV_CTL_EL2](#).ENABLE is 0, the value returned is UNKNOWN.
- If [CNTHV_CTL_EL2](#).ENABLE is 1, the value returned is ([CNTHV_CVAL_EL2](#) - [CNTVCT_EL0](#)).

On a write of this register, [CNTHV_CVAL_EL2](#) is set to ([CNTVCT_EL0](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTHV_CTL_EL2](#).ENABLE is 1, the timer condition is met when ([CNTVCT_EL0](#) - [CNTHV_CVAL_EL2](#)) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTHV_CTL_EL2](#).ISTATUS is set to 1.
- If [CNTHV_CTL_EL2](#).IMASK is 0, an interrupt is generated.

When [CNTHV_CTL_EL2](#).ENABLE is 0, the timer condition is not met, but [CNTVCT_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 CNTHV_TVAL_EL2

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic CNTHV_TVAL_EL2 or CNTV_TVAL_EL0 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>, CNTHV_TVAL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	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
    if CNTHV_CTL_EL2.ENABLE == '0' then
        return bits(64) UNKNOWNCNTHV_TVAL_EL2;
    else
        return CNTHV_CVAL_EL2 - PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    if CNTHV_CTL_EL2.ENABLE == '0' then
        return bits(64) UNKNOWNCNTHV_TVAL_EL2;
    else
        return CNTHV_CVAL_EL2 - PhysicalCountInt();

```

MSR CNTHV_TVAL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	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
    CNTHV_CVAL_EL2CNTHV_TVAL_EL2 = SignExtend((X[t]<31:0>), 64) + PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    CNTHV_CVAL_EL2CNTHV_TVAL_EL2 = SignExtend((X[t]<31:0>), 64) + PhysicalCountInt();

```

MRS <Xt>, CNTV_TVAL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
        IsFeatureImplemented(FEAT_SEL2) then
        if CNTHVS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWN<CNTHVS_TVAL_EL2>;
        else
            return CNTHVS_CVAL_EL2 - PhysicalCountInt();
        elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
            if CNTHV_CTL_EL2.ENABLE == '0' then
                return bits(64) UNKNOWN<CNTHV_TVAL_EL2>;
            else
                return CNTHV_CVAL_EL2 - PhysicalCountInt();
            elseif HaveEL(EL2) && (!EL2Enabled() || HCR_EL2.<E2H,TGE> != '11') then
                if CNTV_CTL_EL0.ENABLE == '0' then
                    return bits(64) UNKNOWN;
                else
                    return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);
            else
                if CNTV_CTL_EL0.ENABLE == '0' then
                    return bits(64) UNKNOWN<CNTV_TVAL_EL0>;
                else
                    return CNTV_CVAL_EL0 - PhysicalCountInt();
            elseif PSTATE.EL == EL1 then
                if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
                    AArch64.SystemAccessTrap(EL2, 0x18);
                elseif HaveEL(EL2) then
                    if CNTV_CTL_EL0.ENABLE == '0' then
                        return bits(64) UNKNOWN;
                    else
                        return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);
                else
                    if CNTV_CTL_EL0.ENABLE == '0' then
                        return bits(64) UNKNOWN<CNTV_TVAL_EL0>;
                    else
                        return CNTV_CVAL_EL0 - PhysicalCountInt();
            elseif PSTATE.EL == EL2 then
                if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
                    if CNTHVS_CTL_EL2.ENABLE == '0' then
                        return bits(64) UNKNOWN<CNTHVS_TVAL_EL2>;
                    else
                        return CNTHVS_CVAL_EL2 - PhysicalCountInt();
                elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
                    if CNTHV_CTL_EL2.ENABLE == '0' then
                        return bits(64) UNKNOWN<CNTHV_TVAL_EL2>;
                    else
                        return CNTHV_CVAL_EL2 - PhysicalCountInt();
                elseif HCR_EL2.E2H == '0' then
                    if CNTV_CTL_EL0.ENABLE == '0' then
                        return bits(64) UNKNOWN;
                    else
                        return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);
                else
                    if CNTV_CTL_EL0.ENABLE == '0' then
                        return bits(64) UNKNOWN<CNTV_TVAL_EL0>;
                    else
                        return CNTV_CVAL_EL0 - PhysicalCountInt();
            elseif PSTATE.EL == EL3 then
                if CNTV_CTL_EL0.ENABLE == '0' then
                    return bits(64) UNKNOWN<CNTV_TVAL_EL0>;
                elseif HaveEL(EL2) && !ELUsingAArch32(EL2) then
                    return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);

```

```

elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
    return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTV0FF);
else
    return CNTV_CVAL_EL0 - PhysicalCountInt();

```

MSR CNTV_TVAL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2CNTHVS_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        CNTHV_CVAL_EL2CNTHV_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
    elseif HaveEL(EL2) && (!EL2Enabled() || HCR_EL2.<E2H,TGE> != '11') then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF_EL2;
    else
        CNTV_CVAL_EL0CNTV_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL2) then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF_EL2;
    else
        CNTV_CVAL_EL0CNTV_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2CNTHVS_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        CNTHV_CVAL_EL2CNTHV_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
    elseif HCR_EL2.E2H == '0' then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF_EL2;
    else
        CNTV_CVAL_EL0CNTV_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
elseif PSTATE.EL == EL3 then
    ifCNTV_TVAL_EL0 HaveEL(EL2) && !ELUsingAArch32(EL2) then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF_EL2;
    elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF;
    else
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTHVS_CTL_EL2, Counter-timer Secure Virtual Timer Control register (EL2)

The CNTHVS_CTL_EL2 characteristics are:

Purpose

Control register for the Secure EL2 virtual timer.

Configuration

AArch64 System register CNTHVS_CTL_EL2 bits [31:0] are architecturally mapped to AArch32 System register [CNTHVS_CTL\[31:0\]](#).

This register is present only when **FEAT_SEL2** is implemented and **FEAT_VHE** is implemented. Otherwise, direct accesses to CNTHVS_CTL_EL2 are UNDEFINED.

Attributes

CNTHVS_CTL_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.

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 CNTHVS_CTL_EL2.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 CNTHVS_CTL_EL2.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 [CNTHVS_TVAL_EL2](#) 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 CNTHVS_CTL_EL2

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

MRS <Xt>, CNTHVS_CTL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0100	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsSecure() then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsSecure() then
        UNDEFINED;
    else
        return CNTHVS_CTL_EL2;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        return CNTHVS_CTL_EL2;

```

MSR CNTHVS_CTL_EL2, <Xt>

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

0b11	0b100	0b1110	0b0100	0b001
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsSecure() then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsSecure() then
        UNDEFINED;
    else
        CNTHVS_CTL_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        CNTHVS_CTL_EL2 = X[t];

```

MRS <Xt>, CNTV_CTL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b001

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        return CNTHVS_CTL_EL2;
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        return CNTHV_CTL_EL2;
    else
        return CNTV_CTL_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x170];
    else
        return CNTV_CTL_EL0;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        return CNTHVS_CTL_EL2;
    elsif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        return CNTHV_CTL_EL2;
    else
        return CNTV_CTL_EL0;
elsif PSTATE.EL == EL3 then
    return CNTV_CTL_EL0;

```

MSR CNTV_CTL_EL0, <Xt>

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

0b11	0b011	0b1110	0b0011	0b001
------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CTL_EL2 = X[t];
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        CNTHV_CTL_EL2 = X[t];
    else
        CNTV_CTL_EL0 = X[t];
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x170] = X[t];
    else
        CNTV_CTL_EL0 = X[t];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CTL_EL2 = X[t];
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        CNTHV_CTL_EL2 = X[t];
    else
        CNTV_CTL_EL0 = X[t];
elseif PSTATE.EL == EL3 then
    CNTV_CTL_EL0 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTHVS_CVAL_EL2, Counter-timer Secure Virtual Timer CompareValue register (EL2)

The CNTHVS_CVAL_EL2 characteristics are:

Purpose

Holds the compare value for the Secure EL2 virtual timer.

Configuration

AArch64 System register CNTHVS_CVAL_EL2 bits [63:0] are architecturally mapped to AArch32 System register [CNTHVS_CVAL\[63:0\]](#).

This register is present only when [FEAT_SEL2](#) is implemented and [FEAT_VHE](#) is implemented. Otherwise, direct accesses to CNTHVS_CVAL_EL2 are UNDEFINED.

Attributes

CNTHVS_CVAL_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
																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 EL2 virtual timer CompareValue.

When [CNTHVS_CTL_EL2.ENABLE](#) is 1, the timer condition is met when ([CNTVCT_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:

- [CNTHVS_CTL_EL2.ISTATUS](#) is set to 1.
- If [CNTHVS_CTL_EL2.IMASK](#) is 0, an interrupt is generated.

When [CNTHVS_CTL_EL2.ENABLE](#) is 0, the timer condition is not met, but [CNTVCT_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 CNTHVS_CVAL_EL2

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

MRS <Xt>, CNTHVS_CVAL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0100	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsSecure() then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsSecure() then
        UNDEFINED;
    else
        return CNTHVS_CVAL_EL2;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        return CNTHVS_CVAL_EL2;

```

MSR CNTHVS_CVAL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0100	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsSecure() then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsSecure() then
        UNDEFINED;
    else
        CNTHVS_CVAL_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        CNTHVS_CVAL_EL2 = X[t];

```

MRS <Xt>, CNTV_CVAL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b010

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        return CNTHVS_CVAL_EL2;
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        return CNTHV_CVAL_EL2;
    else
        return CNTV_CVAL_EL0;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x168];
    else
        return CNTV_CVAL_EL0;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        return CNTHVS_CVAL_EL2;
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        return CNTHV_CVAL_EL2;
    else
        return CNTV_CVAL_EL0;
elseif PSTATE.EL == EL3 then
    return CNTV_CVAL_EL0;

```

MSR CNTV_CVAL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b010

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2 = X[t];
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        CNTHV_CVAL_EL2 = X[t];
    else
        CNTV_CVAL_EL0 = X[t];
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x168] = X[t];
    else
        CNTV_CVAL_EL0 = X[t];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2 = X[t];
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        CNTHV_CVAL_EL2 = X[t];
    else
        CNTV_CVAL_EL0 = X[t];
elseif PSTATE.EL == EL3 then
    CNTV_CVAL_EL0 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdff36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTHVS_TVAL_EL2, Counter-timer Secure Virtual Timer TimerValue register (EL2)

The CNTHVS_TVAL_EL2 characteristics are:

Purpose

Holds the timer value for the Secure EL2 virtual timer.

Configuration

AArch64 System register CNTHVS_TVAL_EL2 bits [31:0] are architecturally mapped to AArch32 System register [CNTHVS_TVAL\[31:0\]](#).

This register is present only when [FEAT_SEL2](#) is implemented and [FEAT_VHE](#) is implemented. Otherwise, direct accesses to CNTHVS_TVAL_EL2 are UNDEFINED.

Attributes

CNTHVS_TVAL_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																															
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 EL2 virtual timer.

On a read of this register:

- If [CNTHVS_CTL_EL2.ENABLE](#) is 0, the value returned is UNKNOWN.
- If [CNTHVS_CTL_EL2.ENABLE](#) is 1, the value returned is ([CNTHVS_CVAL_EL2](#) - [CNTVCT_ELO](#)).

On a write of this register, [CNTHVS_CVAL_EL2](#) is set to ([CNTVCT_ELO](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTHVS_CTL_EL2.ENABLE](#) is 1, the timer condition is met when (([CNTVCT_ELO](#) - [CNTHVS_CVAL_EL2](#)) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTHVS_CTL_EL2.ISTATUS](#) is set to 1.
- If [CNTHVS_CTL_EL2.IMASK](#) is 0, an interrupt is generated.

When [CNTHVS_CTL_EL2.ENABLE](#) is 0, the timer condition is not met, but [CNTVCT_ELO](#) 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 CNTHVS_TVAL_EL2

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

MRS <Xt>, CNTHVS_TVAL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsSecure() then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsSecure() then
        UNDEFINED;
    else
        if CNTHVS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWN_CNTHVS_TVAL_EL2;
        else
            return CNTHVS_CVAL_EL2 - PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        if CNTHVS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWN_CNTHVS_TVAL_EL2;
        else
            return CNTHVS_CVAL_EL2 - PhysicalCountInt();

```

MSR CNTHVS_TVAL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsSecure() then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsSecure() then
        UNDEFINED;
    else
        CNTHVS_CVAL_EL2_CNTHVS_TVAL_EL2 = SignExtend((X[t]<31:0>), 64) + PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        CNTHVS_CVAL_EL2_CNTHVS_TVAL_EL2 = SignExtend((X[t]<31:0>), 64) + PhysicalCountInt();

```

MRS <Xt>, CNTV_TVAL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        if CNTHVS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHVS_TVAL_EL2;
        else
            return CNTHVS_CVAL_EL2 - PhysicalCountInt();
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        if CNTHV_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHV_TVAL_EL2;
        else
            return CNTHV_CVAL_EL2 - PhysicalCountInt();
    elseif HaveEL(EL2) && (!EL2Enabled() || HCR_EL2.<E2H,TGE> != '11') then
        if CNTV_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWN;
        else
            return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);
    else
        if CNTV_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTV_TVAL_EL0;
        else
            return CNTV_CVAL_EL0 - PhysicalCountInt();
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL2) then
        if CNTV_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWN;
        else
            return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);
    else
        if CNTV_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTV_TVAL_EL0;
        else
            return CNTV_CVAL_EL0 - PhysicalCountInt();
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        if CNTHVS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHVS_TVAL_EL2;
        else
            return CNTHVS_CVAL_EL2 - PhysicalCountInt();
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        if CNTHV_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHV_TVAL_EL2;
        else
            return CNTHV_CVAL_EL2 - PhysicalCountInt();
    elseif HCR_EL2.E2H == '0' then
        if CNTV_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWN;
        else
            return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);
    else
        if CNTV_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTV_TVAL_EL0;
        else
            return CNTV_CVAL_EL0 - PhysicalCountInt();
elseif PSTATE.EL == EL3 then
    if CNTV_CTL_EL0.ENABLE == '0' then
        return bits(64) UNKNOWNCNTV_TVAL_EL0;
    elseif HaveEL(EL2) && !ELUsingAArch32(EL2) then
        return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);

```



```

elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
    return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTV0FF);
else
    return CNTV_CVAL_EL0 - PhysicalCountInt();

```

MSR CNTV_TVAL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2CNTHVS_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();];
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        CNTHV_CVAL_EL2CNTHV_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();];
    elseif HaveEL(EL2) && (!EL2Enabled() || HCR_EL2.<E2H,TGE> != '11') then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF_EL2;
    else
        CNTV_CVAL_EL0CNTV_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();];
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL2) then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF_EL2;
    else
        CNTV_CVAL_EL0CNTV_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2CNTHVS_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();];
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        CNTHV_CVAL_EL2CNTHV_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();];
    elseif HCR_EL2.E2H == '0' then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF_EL2;
    else
        CNTV_CVAL_EL0CNTV_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();];
elseif PSTATE.EL == EL3 then
    ifCNTV_TVAL_EL0 HaveEL(EL2) && !ELUsingAArch32(EL2) then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF_EL2;
    elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF;
    else
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTKCTL_EL1, Counter-timer Kernel Control register

The CNTKCTL_EL1 characteristics are:

Purpose

When FEAT_VHE is not implemented, or when [HCR_EL2](#).{E2H, TGE} is not {1, 1}, this register controls the generation of an event stream from the virtual counter, and access from EL0 to the physical counter, virtual counter, EL1 physical timers, and the virtual timer.

When FEAT_VHE is implemented and [HCR_EL2](#).{E2H, TGE} is {1, 1}, this register does not cause any event stream from the virtual counter to be generated, and does not control access to the counters and timers. The access to counters and timers at EL0 is controlled by [CNTHCTL_EL2](#).

Configuration

AArch64 System register CNTKCTL_EL1 bits [31:0] are architecturally mapped to AArch32 System register [CNTKCTL\[31:0\]](#).

Attributes

CNTKCTL_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																EVNTIS	RES0								ELOPTEN	ELOVTEN	EVNTI	EVNTDIR	EVNTEN	ELOVCTEN	ELOPCTEN
31	30	29	28	27	26	25	24	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 [63:18]

Reserved, RES0.

EVNTIS, bit [17]

When FEAT_ECV is implemented:

Controls the scale of the generation of the event stream.

EVNTIS	Meaning
0b0	The CNTKCTL_EL1.EVNTI field applies to CNTVCT_ELO [15:0].
0b1	The CNTKCTL_EL1.EVNTI field applies to CNTVCT_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.

Bits [16:10]

Reserved, RES0.

ELOPTEN, bit [9]

Traps EL0 accesses to the physical timer registers to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1, as follows:

- In AArch64 state, the following registers are trapped, reported using EC syndrome value 0x18:
 - [CNTP_CTL_EL0](#), [CNTP_CVAL_EL0](#), and [CNTP_TVAL_EL0](#).
- In AArch32 state, MRC and MCR accesses to the following registers are trapped, reported using EC syndrome value 0x03, MRRC and MCRR accesses are trapped, reported using EC syndrome value 0x04:
 - [CNTP_CTL](#), [CNTP_CVAL](#), [CNTP_TVAL](#).

ELOPTEN	Meaning
0b0	EL0 accesses to the physical timer registers are trapped to EL1.
0b1	This control does not cause any instructions to be trapped.

When FEAT_VHE is implemented and [HCR_EL2](#).{E2H, TGE} is {1, 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.

ELOVTEN, bit [8]

Traps EL0 accesses to the virtual timer registers to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2.TGE](#) is 1, as follows:

- In AArch64 state, accesses to the following registers are trapped, reported using EC syndrome value 0x18:
 - [CNTV_CTL_EL0](#), [CNTV_CVAL_EL0](#), and [CNTV_TVAL_EL0](#).
- In AArch32 state, MRC and MCR accesses to the following registers are trapped and reported using EC syndrome value 0x03, MRRC and MCRR accesses are trapped using EC syndrome value 0x04:
 - [CNTV_CTL](#), [CNTV_CVAL](#), and [CNTV_TVAL](#).

ELOVTEN	Meaning
0b0	EL0 accesses to the virtual timer registers are trapped.
0b1	This control does not cause any instructions to be trapped.

When FEAT_VHE is implemented and [HCR_EL2](#).{E2H, TGE} is {1, 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.

EVNTI, bits [7:4]

Selects which bit of ~~the counter register~~ [CNTVCT_EL0](#), as seen from EL1, is the trigger for the event stream generated from that counter, when that stream is enabled.

If FEAT_ECV is implemented, and CNTKCTL_EL1.EVNTIS is 1, this field selects a trigger bit in the range 8 to 23 of ~~the counter register~~ [CNTVCT_EL0](#).

Otherwise, this field selects a trigger bit in the range 0 to 15 of ~~the counter register~~ [CNTVCT_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 ~~counter register~~ [CNTVCT_ELO](#) trigger bit, [as seen from EL1 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]

When FEAT_VHE is not implemented, or when [HCR_EL2](#).{E2H, TGE} is not {1, 1}, enables the generation of an event stream from ~~the counter register~~ [CNTVCT_ELO](#) [as seen from EL1.](#)

EVNTEN	Meaning
0b0	Disables the event stream.
0b1	Enables the event stream.

When FEAT_VHE is implemented and [HCR_EL2](#).{E2H, TGE} is {1, 1}, this control does not enable 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]

Traps EL0 accesses to the frequency register and virtual counter register to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2](#).TGE is 1, as follows:

- In AArch64 state, accesses to the following registers are trapped and reported using EC syndrome value 0x18:
 - [CNTVCT_ELO](#) and if [CNTKCTL_EL1](#).ELOPCTEN is 0, [CNTFRQ_ELO](#).
- In AArch32 state, MRC and MCR accesses to the following registers are trapped and reported using EC syndrome value 0x03, MRRC and MCRR accesses are trapped and reported using EC syndrome value 0x04:
 - [CNTVCT](#) and if [CNTKCTL_EL1](#).ELOPCTEN is 0, [CNTFRQ](#).

ELOVCTEN	Meaning
0b0	EL0 accesses to the frequency register and virtual counter registers are trapped.
0b1	This control does not cause any instructions to be trapped.

When FEAT_VHE is implemented and [HCR_EL2](#).{E2H, TGE} is {1, 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.

ELOPCTEN, bit [0]

Traps EL0 accesses to the frequency register and physical counter register to EL1, or to EL2 when it is implemented and enabled for the current Security state and [HCR_EL2](#).TGE is 1, as follows:

- In AArch64 state, the following registers are trapped, reported using EC syndrome value 0x18:
 - [CNTPCT_ELO](#) and if [CNTKCTL_EL1](#).ELOVCTEN is 0, [CNTFRQ_ELO](#).

- In AArch32 state, MCR or MRC accesses the following registers are trapped, reported using EC syndrome value 0x03, MCRR or MRRC accesses are trapped and reported using EC syndrome value 0x04:
 - [CNTPCT](#) and if [CNTKCTL_EL1.EL0VCTEN](#) is 0, [CNTFRQ](#).

EL0PCTEN	Meaning
0b0	EL0 accesses to the frequency register and physical counter register are trapped.
0b1	This control does not cause any instructions to be trapped.

When FEAT_VHE is implemented and [HCR_EL2.{E2H, TGE}](#) is {1, 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.

Accessing CNTKCTL_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic CNTKCTL_EL1 or CNTKCTL_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>, CNTKCTL_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1110	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    return CNTKCTL_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return CNTHCTL_EL2;
    else
        return CNTKCTL_EL1;
elsif PSTATE.EL == EL3 then
    return 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];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        CNTHCTL_EL2 = X[t];
    else
        CNTKCTL_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    CNTKCTL_EL1 = X[t];

```

MRS <Xt>, CNTKCTL_EL12

op0	op1	CRn	CRm	op2
0b11	0b101	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
    if HCR_EL2.E2H == '1' then
        return CNTKCTL_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        return CNTKCTL_EL1;
    else
        UNDEFINED;

```

MSR CNTKCTL_EL12, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	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
    if HCR_EL2.E2H == '1' then
        CNTKCTL_EL1 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        CNTKCTL_EL1 = X[t];
    else
        UNDEFINED;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTP_TVAL_ELO, Counter-timer Physical Timer TimerValue register

The CNTP_TVAL_ELO characteristics are:

Purpose

Holds the timer value for the EL1 physical timer.

Configuration

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

Attributes

CNTP_TVAL_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
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 EL1 physical timer.

On a read of this register:

- If [CNTP_CTL_ELO.ENABLE](#) is 0, the value returned is UNKNOWN.
- If [CNTP_CTL_ELO.ENABLE](#) is 1, the value returned is ([CNTP_CVAL_ELO](#) - [CNTPCT_ELO](#)).

On a write of this register, [CNTP_CVAL_ELO](#) is set to ([CNTPCT_ELO](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTP_CTL_ELO.ENABLE](#) is 1, the timer condition is met when ([CNTPCT_ELO](#) - [CNTP_CVAL_ELO](#)) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTP_CTL_ELO.ISTATUS](#) is set to 1.
- If [CNTP_CTL_ELO.IMASK](#) is 0, an interrupt is generated.

When [CNTP_CTL_ELO.ENABLE](#) is 0, the timer condition is not met, but [CNTPCT_ELO](#) continues to count, so the TimerValue view appears to continue to count down.

Note

The value of [CNTPCT_EL0](#) used in these calculations is the value seen at the Exception Level that the [CNTPCT_EL0](#) register is being read or written from.

The reset behavior of this field is:

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

Accessing CNTP_TVAL_EL0

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic [CNTP_TVAL_EL0](#) or [CNTP_TVAL_EL02](#) 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>, CNTP_TVAL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b000


```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHPS_TVAL_EL2;
        else
            return CNTHPS_CVAL_EL2 - PhysicalCountInt();
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        if CNTHP_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHP_TVAL_EL2;
        else
            return CNTHP_CVAL_EL2 - PhysicalCountInt();
    elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11' then
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWN;
        else
            return CNTP_CVAL_EL0 - (PhysicalCountInt() - CNTPOFF_EL2);
    else
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTP_TVAL_EL0;
        else
            return CNTP_CVAL_EL0 - PhysicalCountInt();
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' then
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWN;
        else
            return CNTP_CVAL_EL0 - (PhysicalCountInt() - CNTPOFF_EL2);
    else
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTP_TVAL_EL0;
        else
            return CNTP_CVAL_EL0 - PhysicalCountInt();
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHPS_TVAL_EL2;
        else
            return CNTHPS_CVAL_EL2 - PhysicalCountInt();
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        if CNTHP_CTL_EL2.ENABLE == '0' then
            return bits(64) UNKNOWNCNTHP_TVAL_EL2;
        else
            return CNTHP_CVAL_EL2 - PhysicalCountInt();
    else
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTP_TVAL_EL0;
        else
            return CNTP_CVAL_EL0 - PhysicalCountInt();
elseif PSTATE.EL == EL3 then
    if CNTP_CTL_EL0.ENABLE == '0' then
        return bits(64) UNKNOWNCNTP_TVAL_EL0;
    else
        return CNTP_CVAL_EL0 - PhysicalCountInt();

```

```
return CNTP_CVAL_EL0 - PhysicalCountInt();
```

MSR CNTP_TVAL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b000

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
            CNTHPS_CVAL_EL2CNTHPS_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
            CNTHP_CVAL_EL2CNTHP_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
        elsif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11' then
            CNTP_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTPOFF_EL2;
        else
            CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' then
            CNTP_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTPOFF_EL2;
        else
            CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
            CNTHPS_CVAL_EL2CNTHPS_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
        elsif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
            CNTHP_CVAL_EL2CNTHP_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
        else
            CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
    elsif PSTATE.EL == EL3 then
        CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();};
```

MRS <Xt>, CNTP_TVAL_EL02

op0	op1	CRn	CRm	op2
0b11	0b101	0b1110	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 HCR_EL2.E2H == '1' then
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTP_TVAL_EL0;
        else
            return CNTP_CVAL_EL0 - PhysicalCountInt();
        else
            UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        if CNTP_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTP_TVAL_EL0;
        else
            return CNTP_CVAL_EL0 - PhysicalCountInt();
        else
            UNDEFINED;

```

MSR CNTP_TVAL_EL02, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1110	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 HCR_EL2.E2H == '1' then
        CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        CNTP_CVAL_EL0CNTP_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
    else
        UNDEFINED;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbdb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTPS_TVAL_EL1, Counter-timer Physical Secure Timer TimerValue register

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

There are no configuration notes.

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_EL0](#) - [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
                return bits(64) UNKNOWN;
            else
                return CNTPS_CVAL_EL1 - (PhysicalCountInt() - CNTPOFF_EL2);
            else
                if CNTPS_CTL_EL1.ENABLE == '0' then
                    return bits(64) UNKNOWN;
                else
                    return CNTPS_CVAL_EL1 - PhysicalCountInt();
                else
                    UNDEFINED;
            elsif PSTATE.EL == EL2 then
                UNDEFINED;
            elsif PSTATE.EL == EL3 then
                if CNTPS_CTL_EL1.ENABLE == '0' then
                    return bits(64) UNKNOWN;
                else
                    return 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]<31:0>),64) + PhysicalCountInt() - CNTPOFF_EL2;
        else
            CNTPS_CVAL_EL1 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
        else
            UNDEFINED;
            elsif PSTATE.EL == EL2 then
                UNDEFINED;
            elsif PSTATE.EL == EL3 then
                CNTPS_CVAL_EL1 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

CNTV_TVAL_ELO, Counter-timer Virtual Timer TimerValue register

The CNTV_TVAL_ELO characteristics are:

Purpose

Holds the timer value for the EL1 virtual timer.

Configuration

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

Attributes

CNTV_TVAL_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
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 EL1 virtual timer.

On a read of this register:

- If [CNTV_CTL_ELO.ENABLE](#) is 0, the value returned is UNKNOWN.
- If [CNTV_CTL_ELO.ENABLE](#) is 1, the value returned is ([CNTV_CVAL_ELO](#) - [CNTVCT_ELO](#)).

On a write of this register, [CNTV_CVAL_ELO](#) is set to ([CNTVCT_ELO](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTV_CTL_ELO.ENABLE](#) is 1, the timer condition is met when ([CNTVCT_ELO](#) - [CNTV_CVAL_ELO](#)) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTV_CTL_ELO.ISTATUS](#) is set to 1.
- If [CNTV_CTL_ELO.IMASK](#) is 0, an interrupt is generated.

When [CNTV_CTL_ELO.ENABLE](#) is 0, the timer condition is not met, but [CNTVCT_ELO](#) 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 CNTV_TVAL_EL0

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic CNTV_TVAL_EL0 or CNTV_TVAL_EL02 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>, CNTV_TVAL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b000


```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
            if CNTHVS_CTL_EL2.ENABLE == '0' then
                return bits(64) UNKNOWN<CNTHVS_TVAL_EL2>;
            else
                return CNTHVS_CVAL_EL2 - PhysicalCountInt();
            elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
                if CNTHV_CTL_EL2.ENABLE == '0' then
                    return bits(64) UNKNOWN<CNTHV_TVAL_EL2>;
                else
                    return CNTHV_CVAL_EL2 - PhysicalCountInt();
                elsif HaveEL(EL2) && (!EL2Enabled() || HCR_EL2.<E2H,TGE> != '11') then
                    if CNTV_CTL_EL0.ENABLE == '0' then
                        return bits(64) UNKNOWN;
                    else
                        return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);
                    else
                        if CNTV_CTL_EL0.ENABLE == '0' then
                            return bits(64) UNKNOWN<CNTV_TVAL_EL0>;
                        else
                            return CNTV_CVAL_EL0 - PhysicalCountInt();
                elsif PSTATE.EL == EL1 then
                    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
                        AArch64.SystemAccessTrap(EL2, 0x18);
                    elsif HaveEL(EL2) then
                        if CNTV_CTL_EL0.ENABLE == '0' then
                            return bits(64) UNKNOWN;
                        else
                            return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);
                    else
                        if CNTV_CTL_EL0.ENABLE == '0' then
                            return bits(64) UNKNOWN<CNTV_TVAL_EL0>;
                        else
                            return CNTV_CVAL_EL0 - PhysicalCountInt();
                elsif PSTATE.EL == EL2 then
                    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
                        if CNTHVS_CTL_EL2.ENABLE == '0' then
                            return bits(64) UNKNOWN<CNTHVS_TVAL_EL2>;
                        else
                            return CNTHVS_CVAL_EL2 - PhysicalCountInt();
                    elsif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
                        if CNTHV_CTL_EL2.ENABLE == '0' then
                            return bits(64) UNKNOWN<CNTHV_TVAL_EL2>;
                        else
                            return CNTHV_CVAL_EL2 - PhysicalCountInt();
                    elsif HCR_EL2.E2H == '0' then
                        if CNTV_CTL_EL0.ENABLE == '0' then
                            return bits(64) UNKNOWN;
                        else
                            return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);
                    else
                        if CNTV_CTL_EL0.ENABLE == '0' then
                            return bits(64) UNKNOWN<CNTV_TVAL_EL0>;
                        else
                            return CNTV_CVAL_EL0 - PhysicalCountInt();
                elsif PSTATE.EL == EL3 then
                    if CNTV_CTL_EL0.ENABLE == '0' then
                        return bits(64) UNKNOWN<CNTV_TVAL_EL0>;
                    elsif HaveEL(EL2) && !ELUsingAArch32(EL2) then
                        return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);

```

```

elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
    return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTV0FF);
else
    return CNTV_CVAL_EL0 - PhysicalCountInt();

```

MSR CNTV_TVAL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2CNTHVS_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        CNTHV_CVAL_EL2CNTHV_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
    elseif HaveEL(EL2) && (!EL2Enabled() || HCR_EL2.<E2H,TGE> != '11') then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF_EL2;
    else
        CNTV_CVAL_EL0CNTV_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL2) then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF_EL2;
    else
        CNTV_CVAL_EL0CNTV_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2CNTHVS_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
    elseif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1' then
        CNTHV_CVAL_EL2CNTHV_TVAL_EL2 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
    elseif HCR_EL2.E2H == '0' then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF_EL2;
    else
        CNTV_CVAL_EL0CNTV_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();
elseif PSTATE.EL == EL3 then
    ifCNTV_TVAL_EL0 HaveEL(EL2) && !ELUsingAArch32(EL2) then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF_EL2;
    elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() - CNTV0FF;
    else
        CNTV_CVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt();

```

MRS <Xt>, CNTV_TVAL_EL02

op0	op1	CRn	CRm	op2
0b11	0b101	0b1110	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
    if HCR_EL2.E2H == '1' then
        if CNTV_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTV_TVAL_EL0;
        else
            return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);
        else
            UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        if CNTV_CTL_EL0.ENABLE == '0' then
            return bits(64) UNKNOWNCNTV_TVAL_EL0;
        else
            return CNTV_CVAL_EL0 - (PhysicalCountInt() - CNTVOFF_EL2);
        else
            UNDEFINED;

```

MSR CNTV_TVAL_EL02, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	0b1110	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
    if HCR_EL2.E2H == '1' then
        CNTV_CVAL_EL0CNTV_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() -
        CNTVOFF_EL2;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        CNTV_CVAL_EL0CNTV_TVAL_EL0 = SignExtend((X[t]<31:0>),64) + PhysicalCountInt() -
        CNTVOFF_EL2;
    else
        UNDEFINED;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

html diff 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.

Cache prefetch predictions determined by the actions of code in the target execution context or contexts appearing in program order before the instruction cannot **influence** ~~exploitatively control~~ speculative execution 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	Secure.
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}.

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}.

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 ASID 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' && (!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.RestrictPredictionCPP_RCTX(X[t], RestrictType_CachePrefetch);});
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.NV == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.CPPRCTX == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.RestrictPredictionCPP_RCTX(X[t], RestrictType_CachePrefetch);});
    elsif PSTATE.EL == EL2 then
        AArch64.RestrictPredictionCPP_RCTX(X[t], RestrictType_CachePrefetch);});
    elsif PSTATE.EL == EL3 then
        AArch64.RestrictPredictionCPP_RCTX(X[t], RestrictType_CachePrefetch);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The CurrentEL characteristics are:

Purpose

Holds the current Exception level.

Configuration

There are no configuration notes.

Attributes

CurrentEL is 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												EL		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 [63:4]

Reserved, RES0.

EL, bits [3:2]

Current Exception level. Possible values of this field are:

EL	Meaning
0b00	EL0.
0b01	EL1.
0b10	EL2.
0b11	EL3.

When the [HCR_EL2.NV](#) bit is 1, EL1 read accesses to the CurrentEL register return the value of 0b10 in this field.

This field resets to the highest implemented Exception level.

Bits [1:0]

Reserved, RES0.

Accessing CurrentEL

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

MRS <Xt>, CurrentEL

op0	op1	CRn	CRm	op2
0b11	0b000	0b0100	0b0010	0b010


```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        return Zeros(60):'10':Zeros(2);
    else
        return Zeros(60):PSTATE.EL:Zeros(2);
elsif PSTATE.EL == EL2 then
    return Zeros(60):PSTATE.EL:Zeros(2);
elsif PSTATE.EL == EL3 then
    return Zeros(60):PSTATE.EL:Zeros(2);

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DAIF, Interrupt Mask Bits

The DAIF characteristics are:

Purpose

Allows access to the interrupt mask bits.

Configuration

There are no configuration notes.

Attributes

DAIF is 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																D	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 [63:10]

Reserved, RES0.

D, bit [9]

Process state D mask. ~~The possible values of this bit are:~~

D	Meaning
0b0	Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are not masked.
0b1	Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are masked.

When the target Exception level of the debug exception is higher than the current Exception level, the exception is not masked by this bit.

The reset behavior of this field is:

- On a Warm reset, this field resets to 1.

A, bit [8]

SError interrupt mask bit. ~~The possible values of this bit are:~~

A	Meaning
0b0	Exception not masked.
0b1	Exception masked.

The reset behavior of this field is:

- On a Warm reset, this field resets to 1.

I, bit [7]

IRQ mask bit. **The possible values of this bit are:**

I	Meaning
0b0	Exception not masked.
0b1	Exception masked.

The reset behavior of this field is:

- On a Warm reset, this field resets to 1.

F, bit [6]

FIQ mask bit. **The possible values of this bit are:**

F	Meaning
0b0	Exception not masked.
0b1	Exception masked.

The reset behavior of this field is:

- On a Warm reset, this field resets to 1.

Bits [5:0]

Reserved, RES0.

Accessing DAIF

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

MRS <Xt>, DAIF

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b001

```

if PSTATE.EL == EL0 then
    if (EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') || SCTLR_EL1.UMA == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        else
            return Zeros(54):PSTATE.<D,A,I,F>:Zeros(6);
    elsif PSTATE.EL == EL1 then
        return Zeros(54):PSTATE.<D,A,I,F>:Zeros(6);
    elsif PSTATE.EL == EL2 then
        return Zeros(54):PSTATE.<D,A,I,F>:Zeros(6);
    elsif PSTATE.EL == EL3 then
        return Zeros(54):PSTATE.<D,A,I,F>:Zeros(6);

```

MSR DAIF, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0010	0b001

```

if PSTATE.EL == EL0 then
    if (EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') || SCTLR_EL1.UMA == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    else
        PSTATE.<D,A,I,F> = X[t]<9:6>;
elsif PSTATE.EL == EL1 then
    PSTATE.<D,A,I,F> = X[t]<9:6>;
elsif PSTATE.EL == EL2 then
    PSTATE.<D,A,I,F> = X[t]<9:6>;
elsif PSTATE.EL == EL3 then
    PSTATE.<D,A,I,F> = X[t]<9:6>;

```

MSR DAIFSet, #<imm>

op0	op1	CRn	op2
0b00	0b011	0b0100	0b110

MSR DAIFClr, #<imm>

op0	op1	CRn	op2
0b00	0b011	0b0100	0b111

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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
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 CIGDVAC instruction

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.

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 'Permission fault'.

If FEAT_CMOW is implemented, HCR_EL2.{E2H, TGE} is not {1, 1}, SCTL_EL1.CMOW is 1, and EL0 access is enabled, when executed at EL0, the instruction has stage 1 read permission to the VA, but does not have stage 1 write permission to the VA, the instruction generates a stage 1 Permission fault.

If FEAT_CMOW is implemented, HCR_EL2.E2H is 1, SCTL_EL2.CMOW is 1, and EL0 access is enabled, when executed at EL0, the instruction has stage 1 read permission to the VA but does not have stage 1 write permission to the VA, the instruction generates a stage 1 Permission fault.

If FEAT_CMOW is implemented, HCRX_EL2.CMOW is 1, and EL1 or EL0 access is enabled, when executed at EL1 or EL0, the instruction has stage 2 read permission to the VA but does not have stage 2 write permission to the VA, the instruction generates a stage 2 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') && 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.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGITR_EL2.DCCIVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTL_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t], 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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCIVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t], CacheType_Data_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t], CacheType_Data_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t], CacheType_Data_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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
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 CIGVAC instruction

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.

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 'Permission fault'.

If FEAT_CMOW is implemented, HCR_EL2.{E2H, TGE} is not {1, 1}, SCTL_EL1.CMOW is 1, and EL0 access is enabled, when executed at EL0, the instruction has stage 1 read permission to the VA, but does not have stage 1 write permission to the VA, the instruction generates a stage 1 Permission fault.

If FEAT_CMOW is implemented, HCR_EL2.E2H is 1, SCTL_EL2.CMOW is 1, and EL0 access is enabled, when executed at EL0, the instruction has stage 1 read permission to the VA but does not have stage 1 write permission to the VA, the instruction generates a stage 1 Permission fault.

If FEAT_CMOW is implemented, HCRX_EL2.CMOW is 1, and EL1 or EL0 access is enabled, when executed at EL1 or EL0, the instruction has stage 2 read permission to the VA but does not have stage 2 write permission to the VA, the instruction generates a stage 2 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') && 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' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGITR_EL2.DCCIVAC == '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], CacheType_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCIVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t], CacheType_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t], CacheType_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t], CacheType_Tag, CacheOp_CleanInvalidate, CacheOpScope_PoC);

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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.

When FEAT_MTE2 is implemented, this instruction might clean and invalidate Allocation Tags from caches.

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

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.

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 'Permission fault'.

When FEAT_CMOW is implemented, [HCR_EL2](#).{E2H, TGE} is not {1, 1}, [SCTLR_EL1](#).CMOW is 1, and EL0 access is implemented, when executed at EL0, the instruction has stage 1 read permission to the VA, but does not have stage 1 write permission to the VA, the instruction generates a stage 1 Permission fault.

When FEAT_CMOW is implemented, [HCR_EL2](#).E2H is 1, [SCTLR_EL2](#).CMOW is 1, and EL0 access is implemented, when executed at EL0, the instruction has stage 1 read permission to the VA but does not have stage 1 write permission to the VA, the instruction generates a stage 1 Permission fault.

When FEAT_CMOW is implemented, [HCRX_EL2](#).CMOW is 1, and EL1 or EL0 access is enabled, when executed at EL1 or EL0, the instruction has stage 2 read permission to the VA but does not have stage 2 write permission to the VA, the instruction generates a stage 2 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') && 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' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGITR_EL2.DCCIVAC == '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], CacheType_Data, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TPCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DCCIVAC == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.DC(X[t], CacheType_Data, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL2 then
        AArch64.DC(X[t], CacheType_Data, CacheOp_CleanInvalidate, CacheOpScope_PoC);
    elsif PSTATE.EL == EL3 then
        AArch64.DC(X[t], CacheType_Data, CacheOp_CleanInvalidate, CacheOpScope_PoC);

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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, DSPSR_EL0.E is RES0. If the implementation does not support little-endian operation, DSPSR_EL0.E is RES1. On exiting Debug state, if the implementation does not support big-endian operation at the Exception level being returned to, DSPSR_EL0.E is RES0, and if the implementation does not support little-endian operation at the Exception level being returned to, DSPSR_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.

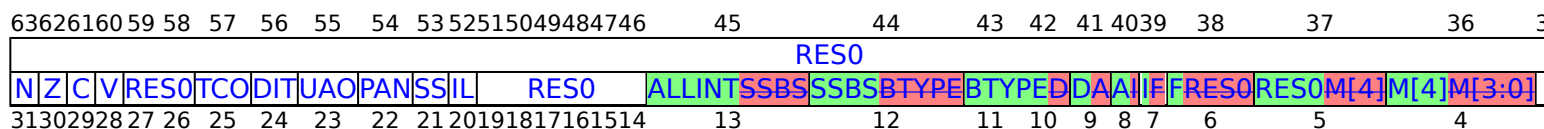
M[3:0]	Meaning
0b0000	User.
0b0001	FIQ.
0b0010	IRQ.
0b0011	Supervisor.
0b0110	Monitor.
0b0111	Abort.
0b1010	Hyp.
0b1011	Undefined.
0b1111	System.

Other values are reserved. If DSPSR_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 [63:32]

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_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 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 [19:14:13]

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.

BTYPE, 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]

Error 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>, DSPSR_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0101	0b000

```
if !Halted() then
    UNDEFINED;
else
    return DSPSR_EL0;
```

MSR DSPSR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0101	0b000

```
if !Halted() then
    UNDEFINED;
else
    DSPSR_EL0 = X[t];
```

3020/09/2021 14:12:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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				NSENS	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	Secure.
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}.

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}.

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 ASID 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' && (!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.RestrictPredictionDVP_RCTX(X[t], RestrictType_DataValue);});
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.NV == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.DVPRCTX == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.RestrictPredictionDVP_RCTX(X[t], RestrictType_DataValue);});
    elsif PSTATE.EL == EL2 then
        AArch64.RestrictPredictionDVP_RCTX(X[t], RestrictType_DataValue);});
    elsif PSTATE.EL == EL3 then
        AArch64.RestrictPredictionDVP_RCTX(X[t], RestrictType_DataValue);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(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 [63:37]

Reserved, RES0.

ISS2, bits [36:32]

When FEAT_LS64 is implemented:

If a memory access generated by an ST64BV or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

For any other Data Abort, this field is RES0.

Otherwise:

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 as described in 'The EC used to report an exception routed to EL2 because HCR_EL2.TGE is 1'.	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,	ISS encoding for an exception	When FEAT_LS64

	ST64B, ST64BV, or ST64BV0 instruction.	from an LD64B or ST64B* instruction	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.	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
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, CPTER_EL2.ZEN, CPTER_EL2.TZ, or CPTER_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	ISS encoding for an exception from a TSTART instruction	When FEAT_TME is implemented

	<p>== 0, at EL2 when SCTLR_EL2.TME == 0 or at EL3 when SCTLR_EL3.TME == 0.</p>		
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 from a lower Exception level.	ISS encoding for an exception	

	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.	from a Data Abort	
0b100101	Data Abort 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	ISS encoding for an exception from a trapped floating-point exception	When AArch64 is supported

	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.		
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

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]

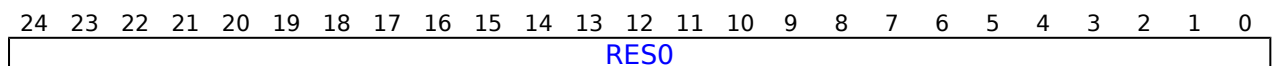
Instruction Specific Syndrome. Architecturally, this field can be defined independently for each defined Exception class. However, in practice, some ISS encodings are used for more than one Exception class.

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. See 'Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32'.
- 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.

When FEAT_WFxT2 is not implemented, RV is set to 0 on a trap on WFET or WFIT.

This field is set to 1 on a trap on WFET or WFIT.

When FEAT_WFxT2 is implemented, RV 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:

- [CNTKCTL_EL1](#).{ELOPTEN, EL0VTEN, ELOPCTEN, 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.
- [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.
- For information on other traps using EC value 0b000011, see 'Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32'.
- If FEAT_FGT is implemented, MCR or MRC access to some registers at EL0, trapped to EL2.

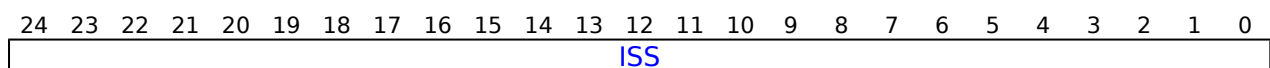
The following fields describe configuration settings for generating exceptions that are reported using EC value 0b000101:

- [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
0b000000000000000000000000	ST64BV instruction trapped.
0b000000000000000000000001	ST64BV0 instruction trapped.
0b000000000000000000000010	LD64B or ST64B instruction trapped.

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 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.

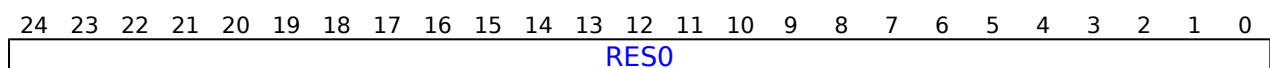
Bits [19:0]

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.

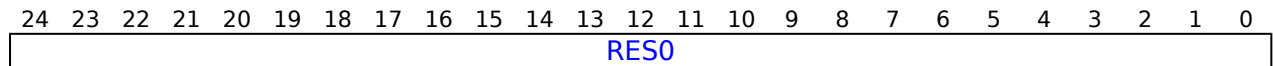
Bits [24:0]

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 an exception from an Illegal Execution state, or a PC or SP alignment fault



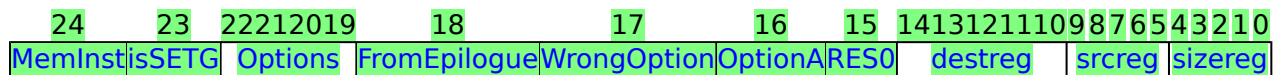
Bits [24:0]

Reserved, RES0.

There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. For more information about these exceptions, see 'The Illegal Execution state exception' and '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.

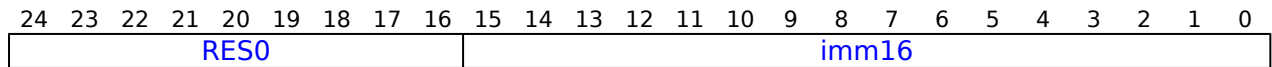
sizereg, bits [4:0]

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



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.

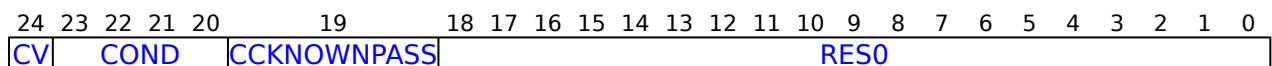
In AArch32 state, the HVC instruction is unconditional, and a conditional SVC instruction generates an exception only if it passes its condition code check. Therefore, the syndrome information for these exceptions does not require conditionality information.

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.

'System calls' describes the case where these exceptions are trapped to EL3.

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.

'System calls' describes the case where these exceptions are trapped to EL3.

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:

- [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.
- [CNTCTL_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.
- [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
 - [HDFGRTR_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 [SMPRIMAP_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 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						

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.	
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 When FEAT_RAS is not 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
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'.

Note

Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

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 the [SVCR](#), and the SME System registers [SMCR_EL1](#), [SMCR_EL2](#), [SMCR_EL3](#), [SMPRI_EL1](#), and [SMPRMAP_EL2](#).

Bits [24:32]

Reserved, RES0.

SMTC, bits [21:0]

SME Trap Code. Identifies the reason for instruction trapping.

SMTC	Meaning
0b0000b000	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.
0b0010b001	Advanced SIMD, SVE, or SVE2 instruction trapped because PSTATE.SM is 1.
0b0100b010	SME instruction trapped because PSTATE.SM is 0.
0b0110b011	SME instruction trapped because PSTATE.ZA is 0.

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 the [SVCR](#) and [SMCR_EL1](#) System registers at EL1 and EL0, trapped to [EL1](#) or [EL2](#).[EL1](#).
- [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 the [SVCR](#), [SMCR_EL1](#), and [SMCR_EL2](#) System registers 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 the [SVCR](#) and other SME System registers from all Exception levels and any Security state, to EL3. [SMCR_EL1](#), [SMCR_EL2](#), [SMCR_EL3](#) from all Exception levels and any Security state, 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		S2PTW		InD	GPCSC			VNCR			RES0		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.
0b000001	GPT address size fault at level 1.
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 [12:11]

Reserved, RES0.

Bits [10:9]

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. **The possible values of this bit are:**

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'.

Note

Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

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			SF	AR	VNCR	Bits[12:11]			FnV	EACM	S1PTW	WnR	DFSC								

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV or 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 ST64BV, ST64BV0, ST64B, or LD64B 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 ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault. ISV is 0 for all other faults reported in ESR_EL1 or ESR_EL3.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

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_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.

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.

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 ST64BV, ST64BV0, ST64B, or LD64B 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 ST64BV, ST64BV0, ST64B, or LD64B 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.

SF, bit [15]

When ISV == 1:

Width of the register accessed by the instruction is Sixty-Four.

SF	Meaning
0b0	Instruction loads/stores a 32-bit wide register.
0b1	Instruction loads/stores a 64-bit wide 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 ST64BV, ST64BV0, ST64B, or LD64B 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.

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 ST64BV, ST64BV0, ST64B, or LD64B 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.

SET, bits [12:11]**When FEAT_RAS is implemented and FEAT_LS64 is not implemented:**

Synchronous Error Type. When DFSC is 0b010000, 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.

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.

When FEAT_LS64 is implemented:

Load/Store Type. Used when an LD64B, ST64B, ST64BV, or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

LST	Meaning
0b01	An ST64BV instruction generated the Data Abort.
0b10	An LD64B or ST64B instruction generated the Data Abort.
0b11	An ST64BV0 instruction generated the Data Abort.

All other values are reserved.

This field is valid only if the DFSC code is 0b110101. 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 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
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.	
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 level associated with MMU faults'.

Note

Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

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:

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.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

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:

Asynchronous Error Type.

When DFSC is 0b010001, 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.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

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:**

External abort type. When DFSC is 0b010001, provides an IMPLEMENTATION DEFINED classification of External aborts.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

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										RES0VNCR		RES0			CMRES0WnR		DFSC							

Bits [24:15]

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 [12:9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Watchpoint exception came from a cache maintenance or address translation 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 either the execution of a cache maintenance instruction or by a synchronous Watchpoint exception on the execution of an address translation 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 and address translation 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 ERETA* 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

24	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]

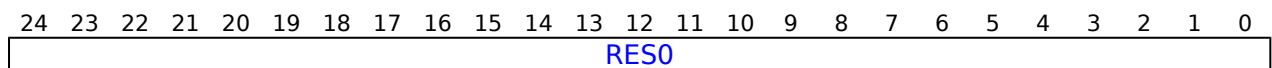
Reserved, RES0.

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

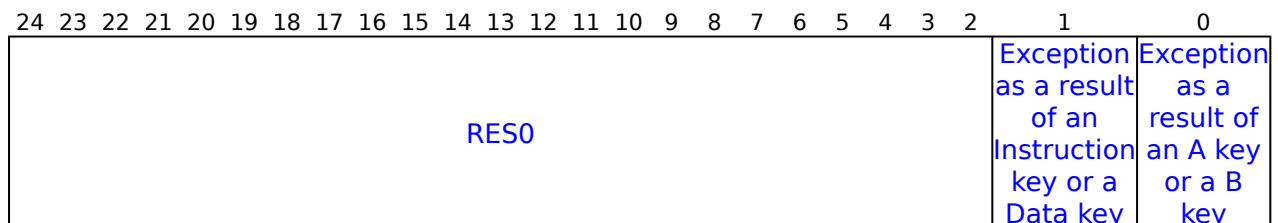
**Bits [24:0]**

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_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() && (!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
        return NVMem[0x138];
    else
        return ESR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return ESR_EL2;
    else
        return ESR_EL1;
elsif PSTATE.EL == EL3 then
    return 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() && (!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];
    else
        ESR_EL1 = X[t];
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            ESR_EL2 = X[t];
        else
            ESR_EL1 = X[t];
    elsif PSTATE.EL == EL3 then
        ESR_EL1 = X[t];

```

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
        return 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
            return ESR_EL1;
        else
            UNDEFINED;
    elsif PSTATE.EL == EL3 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
            return 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];
    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];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        ESR_EL1 = X[t];
    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
        return ESR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return ESR_EL2;
elsif PSTATE.EL == EL3 then
    return 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];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    ESR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    ESR_EL2 = X[t];

```

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ESR_EL2, Exception Syndrome Register (EL2)

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 [63:37]

Reserved, RES0.

ISS2, bits [36:32]

When FEAT_LS64 is implemented:

If a memory access generated by an ST64BV or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

For any other Data Abort, this field is RES0.

Otherwise:

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 as described in 'The EC used to report an exception routed to EL2 because HCR_EL2.TGE is 1'.	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 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	Trapped execution of an LD64B, ST64B, ST64BV, or ST64BV0 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 the value of HCR_EL2.TSC is 1.	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
0b011010	Trapped ERET, ERETAA, or ERETAB instruction execution.	ISS encoding for an exception from an ERET, ERETAA, or ERETAB instruction	When FEAT_PAuth is implemented and FEAT_NV 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	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.	authentication failure 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 from a lower Exception level, excluding Data Aborts taken to EL2 as a result of accesses generated associated with VNCR_EL2 as part of nested virtualization support. These Data Aborts might be generated from Exception	ISS encoding for an exception from a Data Abort	

	levels in any Execution state. 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.		
0b100101	Data Abort without a change in Exception level, or Data Aborts 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	ISS encoding for an exception from a trapped floating-point exception	When AArch32 is supported

	floating-point exceptions is IMPLEMENTATION DEFINED.		
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	ISS encoding for an exception from a Watchpoint exception	

	accesses generated associated with VNCR_EL2 as part of nested virtualization support.		
0b111000	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

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]

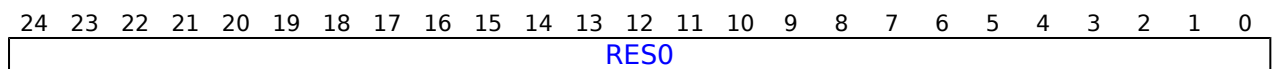
Instruction Specific Syndrome. Architecturally, this field can be defined independently for each defined Exception class. However, in practice, some ISS encodings are used for more than one Exception class.

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. See 'Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32'.
- 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.

When **FEAT_WFXT2** is not implemented, RV is set to 0 on a trap on WFET or WFIT.

This field is set to 1 on a trap on WFET or WFIT.

When **FEAT_WFXT2** is implemented, RV 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:

- [CNTKCTL_EL1](#).{ELOPTEN, EL0VTEN, ELOPCTEN, 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.
- [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.
- For information on other traps using EC value 0b000011, see 'Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32'.
- 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:

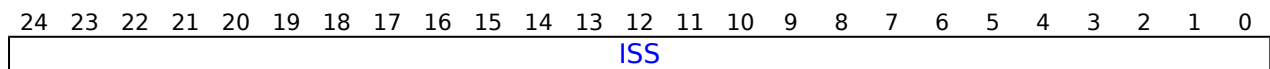
- [CPACR_EL1](#).TTA for accesses to trace registers, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- [MDSCR_EL1](#).TDCC, for accesses to the Debug Communications Channel (DCC) registers at EL0 and EL1 using AArch32 state, MCR or MRC access (coproc == 0b1110) trapped to EL1 or EL2.
- If 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

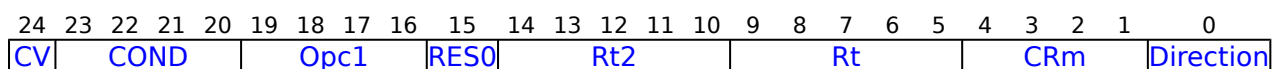


ISS, bits [24:0]

ISS	Meaning
0b000000000000000000000000000000	ST64BV instruction trapped.
0b000000000000000000000000000001	ST64BV0 instruction trapped.
0b000000000000000000000000000010	LD64B or ST64B instruction trapped.

All other values are reserved.

ISS encoding for an exception from an MCRR or MRRC access



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, 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 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]

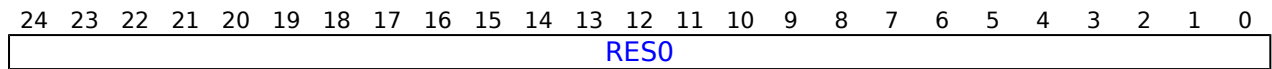
Reserved, RES0.

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.

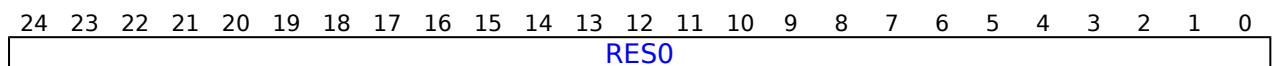
Bits [24:0]

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 an exception from an Illegal Execution state, or a PC or SP alignment fault



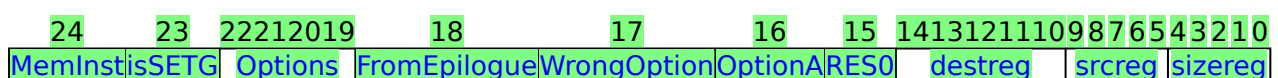
Bits [24:0]

Reserved, RES0.

There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. For more information about these exceptions, see 'The Illegal Execution state exception' and '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 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.

'System calls' describes the case where these exceptions are trapped to EL3.

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.

'System calls' describes the case where these exceptions are trapped to EL3.

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:

- [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](#).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.
- [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.
 - [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 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						

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.	
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 When FEAT_RAS is not 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
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'.

Note

Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

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 the **SVCR**, and the SME System registers **SMCR_EL1**, **SMCR_EL2**, **SMCR_EL3**, **SMPRI_EL1**, and **SMPRMAP_EL2**.

Bits [24:32]

Reserved, RES0.

SMTC, bits [21:0]

SME Trap Code. Identifies the reason for instruction trapping.

SMTC	Meaning
0b0000b000	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.
0b0010b001	Advanced SIMD, SVE, or SVE2 instruction trapped because PSTATE.SM is 1.
0b0100b010	SME instruction trapped because PSTATE.SM is 0.
0b0110b011	SME instruction trapped because PSTATE.ZA is 0.

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 the **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 the **SVCR**, **SMCR_EL1**, and **SMCR_EL2** System registers 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 the **SVCR** and other SME System registers from all Exception levels and any Security state, to EL3. **SMCR_EL1**, **SMCR_EL2**, **SMCR_EL3** from all Exception levels and any Security state, 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		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.
0b000001	GPT address size fault at level 1.
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 [12:11]

Reserved, RES0.

Bits [10:9]

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. **The possible values of this bit are:**

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'.

Note

Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

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			SF	AR	VNCR	Bits[12:11]			FnV	EACM	S1PTW	WnR	DFSC								

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV or 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 ST64BV, ST64BV0, ST64B, or LD64B 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 ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault. ISV is 0 for all other faults reported in ESR_EL1 or ESR_EL3.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

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_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.

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.

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 ST64BV, ST64BV0, ST64B, or LD64B 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 ST64BV, ST64BV0, ST64B, or LD64B 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.

SF, bit [15]

When ISV == 1:

Width of the register accessed by the instruction is Sixty-Four.

SF	Meaning
0b0	Instruction loads/stores a 32-bit wide register.
0b1	Instruction loads/stores a 64-bit wide 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 ST64BV, ST64BV0, ST64B, or LD64B 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.

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 ST64BV, ST64BV0, ST64B, or LD64B 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.

SET, bits [12:11]**When FEAT_RAS is implemented and FEAT_LS64 is not implemented:**

Synchronous Error Type. When DFSC is 0b010000, 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.

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.

When FEAT_LS64 is implemented:

Load/Store Type. Used when an LD64B, ST64B, ST64BV, or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

LST	Meaning
0b01	An ST64BV instruction generated the Data Abort.
0b10	An LD64B or ST64B instruction generated the Data Abort.
0b11	An ST64BV0 instruction generated the Data Abort.

All other values are reserved.

This field is valid only if the DFSC code is 0b110101. 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 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
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.	
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 level associated with MMU faults'.

Note

Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

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:

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.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

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:

Asynchronous Error Type.

When DFSC is 0b010001, 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.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

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:**

External abort type. When DFSC is 0b010001, provides an IMPLEMENTATION DEFINED classification of External aborts.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

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										RES0VNCR		RES0				CMRES0WnR				DFSC					

Bits [24:15]

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 [12:9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Watchpoint exception came from a cache maintenance or address translation 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 either the execution of a cache maintenance instruction or by a synchronous Watchpoint exception on the execution of an address translation 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 and address translation 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 ERETA* 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

24	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]

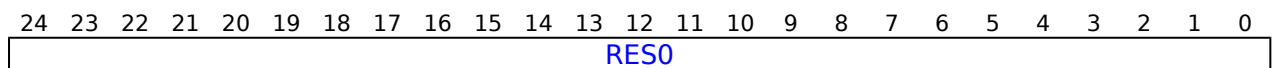
Reserved, RES0.

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

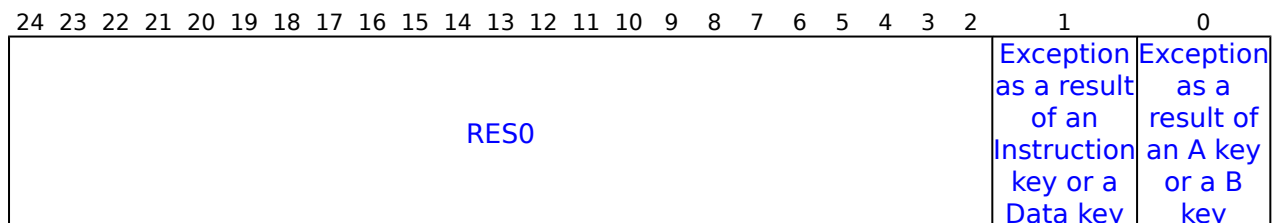
**Bits [24:0]**

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
        return ESR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return ESR_EL2;
elsif PSTATE.EL == EL3 then
    return 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];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    ESR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    ESR_EL2 = X[t];

```

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() && (!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
        return NVMem[0x138];
    else
        return ESR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return ESR_EL2;
    else
        return ESR_EL1;
elsif PSTATE.EL == EL3 then
    return 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() && (!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];
    else
        ESR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        ESR_EL2 = X[t];
    else
        ESR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ESR_EL1 = X[t];

```

3020/09/2021 1412:5236: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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 [63:37]

Reserved, RES0.

ISS2, bits [36:32]

When FEAT_LS64 is implemented:

If a memory access generated by an ST64BV or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault, then this field holds register specifier, Xs.

For any other Data Abort, this field is RES0.

Otherwise:

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 as described in 'The EC used to report an exception routed to EL2 because HCR_EL2.TGE is 1'.	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	ISS encoding for an exception from a Pointer	When FEAT_PAuth

	instruction because HCR_EL2.API == 0 SCR_EL3.API == 0 .	Authentication instruction when HCR_EL2.API == 0 SCR_EL3.API == 0	is implemented
0b001010	Trapped execution of an LD64B, ST64B, ST64BV, or ST64BV0 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 ,	ISS encoding for an exception from an access to SVE functionality	When FEAT_SVE is implemented

0b011011	<p>CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ, that is not reported using EC 0b000000. 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>	<p>resulting from CPACR_EL1.ZEN, CPTR_EL2.ZEN, CPTR_EL2.TZ, or CPTR_EL3.EZ ISS encoding for an exception from a TSTART instruction</p>	<p>When FEAT_TME is implemented</p>
0b011100	<p>Exception from a Pointer Authentication instruction authentication failure</p>	<p>ISS encoding for an exception from a Pointer Authentication instruction authentication failure</p>	<p>When FEAT_FPAC is implemented</p>
0b011101	<p>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.</p>	<p>ISS encoding for an exception due to SME functionality</p>	<p>When FEAT_SME is implemented</p>
0b011110	<p>Exception from a Granule Protection Check</p>	<p>ISS encoding for an exception from a Granule Protection Check</p>	<p>When FEAT_RME is implemented</p>
0b011111	<p>IMPLEMENTATION DEFINED exception to EL3.</p>	<p>ISS encoding for an IMPLEMENTATION DEFINED exception to EL3</p>	
0b100000	<p>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.</p>	<p>ISS encoding for an exception from an Instruction Abort</p>	
0b100001	<p>Instruction Abort taken without a change in Exception level.</p>	<p>ISS encoding for an exception from an Instruction Abort</p>	

0b100010	Used for MMU faults generated by instruction accesses and synchronous External aborts, including 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 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 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	ISS encoding for an exception from a trapped floating-point exception	When AArch64 is supported

	exceptions, otherwise it is reserved. Whether a floating-point 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

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.

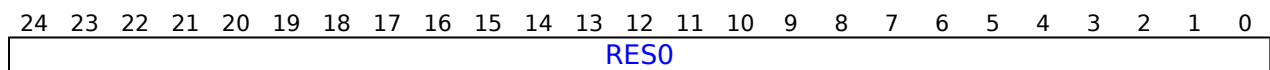
Typically, an ISS encoding has a number of subfields. When an ISS subfield holds a register number, the value returned in that field is the AArch64 view of the register number.

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. See 'Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32'.
- 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.

When **FEAT_WFxT2** is not implemented, RV is set to 0 on a trap on WFET or WFIT.

This field is set to 1 on a trap on WFET or WFIT.

When **FEAT_WFxT2** is implemented, RV 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 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.

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:

- [CNTKCTL_EL1](#).{ELOPTEN, EL0VTEN, ELOPCTEN, 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.
- [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.
- For information on other traps using EC value 0b000011, see 'Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32'.
- 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.
- [MDCR_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
0b00000000000000000000000000000000	ST64BV instruction trapped.
0b00000000000000000000000000000001	ST64BV0 instruction trapped.
0b00000000000000000000000000000010	LD64B or ST64B instruction trapped.

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 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 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 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.

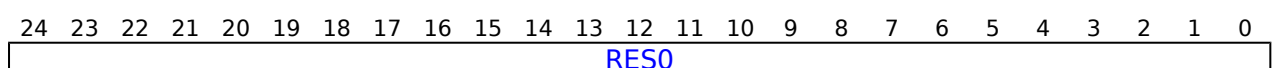
Bits [19:0]

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.

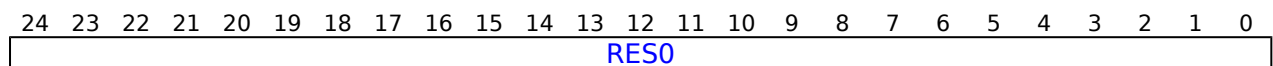
Bits [24:0]

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 an exception from an Illegal Execution state, or a PC or SP alignment fault



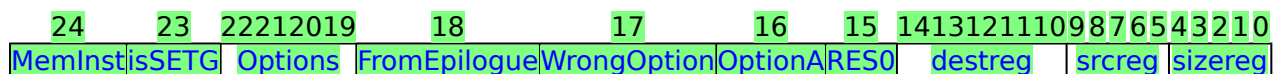
Bits [24:0]

Reserved, RES0.

There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. For more information about these exceptions, see 'The Illegal Execution state exception' and '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:

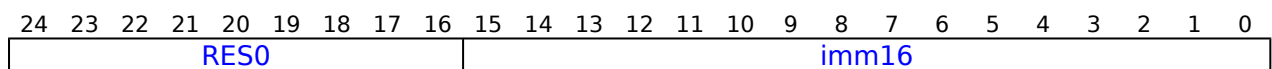
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

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**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.

'System calls' describes the case where these exceptions are trapped to EL3.

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.

'System calls' describes the case where these exceptions are trapped to EL3.

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:

- [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.
- [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.
 - [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 [SMPRIMAP_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 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						

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.	
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 When FEAT_RAS is not 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
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'.

Note

Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

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 the **SVCR**, and the SME System registers **SMCR_EL1**, **SMCR_EL2**, **SMCR_EL3**, **SMPRI_EL1**, and **SMPRMAP_EL2**.

Bits [24:32]

Reserved, RES0.

SMTC, bits [21:0]

SME Trap Code. Identifies the reason for instruction trapping.

SMTC	Meaning
0b0000b000	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.
0b0010b001	Advanced SIMD, SVE, or SVE2 instruction trapped because PSTATE.SM is 1.
0b0100b010	SME instruction trapped because PSTATE.SM is 0.
0b0110b011	SME instruction trapped because PSTATE.ZA is 0.

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 the **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 the **SVCR**, **SMCR_EL1**, and **SMCR_EL2** System registers 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 the **SVCR** and other SME System registers from all Exception levels and any Security state, to EL3. **SMCR_EL1**, **SMCR_EL2**, **SMCR_EL3** from all Exception levels and any Security state, 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		S2PTW		InD	GPCSC			VNCR			RES0	RES0	CM	S1PTW	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.
0b000001	GPT address size fault at level 1.
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 [12:11]

Reserved, RES0.

Bits [10:9]

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. **The possible values of this bit are:**

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'.

Note

Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

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			SF	AR	VNCR	Bits[12:11]			FnV	EACM	S1PTW	WnR	DFSC								

When FEAT_LS64 is implemented, if a memory access generated by an ST64BV or 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 ST64BV, ST64BV0, ST64B, or LD64B 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 ST64BV, ST64BV0, ST64B, or LD64B instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault. ISV is 0 for all other faults reported in ESR_EL1 or ESR_EL3.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

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_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.

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.

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 ST64BV, ST64BV0, ST64B, or LD64B 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:

Syndrone 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 ST64BV, ST64BV0, ST64B, or LD64B 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.

SF, bit [15]

When ISV == 1:

Width of the register accessed by the instruction is Sixty-Four.

SF	Meaning
0b0	Instruction loads/stores a 32-bit wide register.
0b1	Instruction loads/stores a 64-bit wide 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 ST64BV, ST64BV0, ST64B, or LD64B 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.

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 ST64BV, ST64BV0, ST64B, or LD64B 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.

SET, bits [12:11]**When FEAT_RAS is implemented and FEAT_LS64 is not implemented:**

Synchronous Error Type. When DFSC is 0b010000, 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.

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.

When FEAT_LS64 is implemented:

Load/Store Type. Used when an LD64B, ST64B, ST64BV, or ST64BV0 instruction generates a Data Abort for a Translation fault, Access flag fault, or Permission fault.

LST	Meaning
0b01	An ST64BV instruction generated the Data Abort.
0b10	An LD64B or ST64B instruction generated the Data Abort.
0b11	An ST64BV0 instruction generated the Data Abort.

All other values are reserved.

This field is valid only if the DFSC code is 0b110101. 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 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
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.	
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 level associated with MMU faults'.

Note

Because Access flag faults and Permission faults can result only from a Block or Page translation table descriptor, they cannot occur at level 0.

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:

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.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

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:

Asynchronous Error Type.

When DFSC is 0b010001, 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.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

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:**

External abort type. When DFSC is 0b010001, provides an IMPLEMENTATION DEFINED classification of External aborts.

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other errors.

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										RES0VNCR		RES0				CMRES0WnR				DFSC					

Bits [24:15]

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 [12:9]

Reserved, RES0.

CM, bit [8]

Cache maintenance. Indicates whether the Watchpoint exception came from a cache maintenance or address translation 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 either the execution of a cache maintenance instruction or by a synchronous Watchpoint exception on the execution of an address translation 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 and address translation 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 ERETA* 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

24	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]

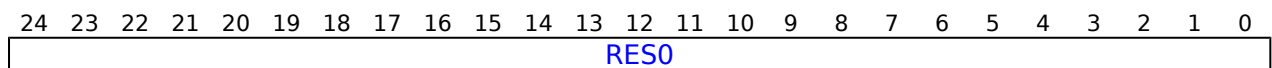
Reserved, RES0.

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

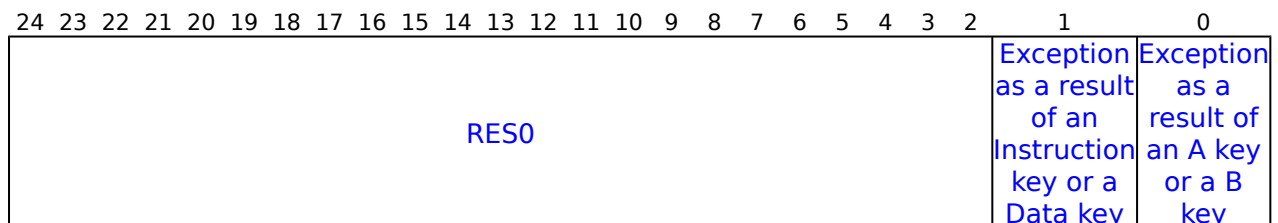
**Bits [24:0]**

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_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
    return 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];
```

(old)

htmldiff from-

(new)

The FAR_EL1 characteristics are:

Purpose

Holds the faulting Virtual Address for all synchronous Instruction or Data Abort, PC alignment fault 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\[31:0\]](#) (NS).

AArch64 System register FAR_EL1 bits [63:32] are architecturally mapped to AArch32 System register [IFAR\[31:0\]](#) (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 the FAR_EL1 is UNKNOWN if [ESR_EL1.FnV](#) is 1.

For all other exceptions taken to EL1, the FAR EL1 is UNKNOWN.

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 that is 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 is CONSTRAINED UNPREDICTABLE.
- The implementation treats such incrementing as setting bit[32] of the virtual address to 1.

For a Data Abort 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'.

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 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 **unaligned** ~~mis-aligned~~ address that crosses a page boundary, the architecture does not prioritize between those different faults.

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() && (!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
        return NVMem[0x220];
    else
        return FAR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return FAR_EL2;
    else
        return FAR_EL1;
elsif PSTATE.EL == EL3 then
    return 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() && (!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];
    else
        FAR_EL1 = X[t];
    elsif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            FAR_EL2 = X[t];
        else
            FAR_EL1 = X[t];
    elsif PSTATE.EL == EL3 then
        FAR_EL1 = X[t];

```

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
        return 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
            return FAR_EL1;
        else
            UNDEFINED;
    elsif PSTATE.EL == EL3 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
            return 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];
    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];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        FAR_EL1 = X[t];
    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
        return FAR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return FAR_EL2;
elsif PSTATE.EL == EL3 then
    return 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];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    FAR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    FAR_EL2 = X[t];

```

(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 or Data Abort, PC alignment fault 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\[31:0\]](#) (S) when EL2 is implemented.

AArch64 System register FAR_EL2 bits [63:32] are architecturally mapped to AArch32 System register [IFAR\[31:0\]](#) (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 the FAR_EL2 is UNKNOWN if [ESR_EL2](#).FnV is 1.

For all other exceptions taken to EL2, the FAR_EL2 is UNKNOWN.

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 that is 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.

For a Data Abort 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'.

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 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 `unaligned` `mis-aligned` address that crosses a page boundary, the architecture does not prioritize between those different faults.

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
        return FAR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return FAR_EL2;
elsif PSTATE.EL == EL3 then
    return 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];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    FAR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    FAR_EL2 = X[t];

```

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() && (!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
        return NVMem[0x220];
    else
        return FAR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return FAR_EL2;
    else
        return FAR_EL1;
elsif PSTATE.EL == EL3 then
    return 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() && (!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];
    else
        FAR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        FAR_EL2 = X[t];
    else
        FAR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    FAR_EL1 = X[t];

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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 or Data Abort 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 the FAR_EL3 is UNKNOWN if [ESR_EL3](#).FnV is 1.

For all other exceptions taken to EL3, the FAR_EL3 is UNKNOWN.

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.

For a Data Abort 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'.

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 **unaligned** ~~mis-aligned~~ address that crosses a page boundary, the architecture does not prioritize between those different faults.

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
    return 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];
```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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

63	62	61	60	59	58	57	56	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								LOGPTSZ				RES0		GPCP	GPC	PGS	SH	ORGN	IRGN	RES0								PPS			
31	30	29	28	27	26	25	24	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.

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**.

Bits [19:18]

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.

ThisThe bitvalue 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.

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, the value of this bitfield 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
    return 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];
```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

HCR_EL2, Hypervisor Configuration Register

The HCR_EL2 characteristics are:

Purpose

Provides configuration controls for virtualization, including defining whether various operations are trapped to EL2.

Configuration

AArch64 System register HCR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HCRI\[31:0\]](#).

AArch64 System register HCR_EL2 bits [63:32] are architecturally mapped to AArch32 System register [HCR2\[31:0\]](#).

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.

Attributes

HCR_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
TWEDEL				TWEDEn	TID5	DCT	ATA	TTLBOS	TTLBIS	EnSCXT	TOCU	AMV	OFFEN	TICAB	TID4	GPF	FIEN	FWB	NV2	ATN
RW	TRVM	HCD	TDZ	TGE	TVM	TTLB	TPU	Bit[23]	TSW	TACR	TIDCP	TSC	TID3	TID2	TID1	TID0	TWE	TWI	DC	I
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11

TWEDEL, bits [63:60]

When FEAT_TWED is implemented:

TWE Delay. A 4-bit unsigned number that, when HCR_EL2.TWEDEn is 1, encodes the minimum delay in taking a trap of WFE* caused by HCR_EL2.TWE as $2^{(TWEDEL + 8)}$ 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 [59]

When FEAT_TWED is implemented:

TWE Delay Enable. Enables a configurable delayed trap of the WFE* instruction caused by HCR_EL2.TWE.

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 HCR_EL2.TWEDEL.

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 EL1 and EL0 access to Allocation Tags.

ATA	Meaning
0b0	Access to Allocation Tags is prevented. Accesses at EL1 to GCR_EL1 , RGSRR_EL1 , TFSRR_EL1 , TFSRR_EL2 , or TFSRRE0_EL1 that are not UNDEFINED are trapped to EL2.
0b1	This control does not prevent access to Allocation Tags.

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, [IC IALLUIS](#).

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 Aborts and Data Aborts 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 [ERXPFPGCDN_EL1](#), [ERXPFPGCTL_EL1](#), and [ERXPFPGF_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 attributes, EL1&0 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. <p>The stage 1 and stage 2 memory types are combined in the manner described in 'Combining the stage 1 and stage 2 attributes, EL1&0 translation regime'.</p>

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.

The possible values are:

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 0x1A 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_PAAuth 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_PAAuth 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_PAAuth 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.

MIOCNCNCE, bit [38]

Mismatched Inner/Outer Cacheable Non-Coherency Enable, for the EL1&0 translation regimes.

MIOCNCNCE	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, the following registers are trapped to EL2 and reported using EC syndrome value 0x18.
 - [SCTLR_EL1](#), [TTBR0_EL1](#), [TTBR1_EL1](#), [TCR_EL1](#), [ESR_EL1](#), [FAR_EL1](#), [AFSR0_EL1](#), [AFSR1_EL1](#), [MAIR_EL1](#), [AMAIR_EL1](#), [CONTEXTIDR_EL1](#).
- If EL1 is using AArch32 state, 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:
 - [SCTLR](#), [TTBR0](#), [TTBR1](#), [TTBCR](#), [TTBCR2](#), [DACR](#), [DFSR](#), [IFSR](#), [DFAR](#), [IFAR](#), [ADFSR](#), [AIFSR](#), [PRRR](#), [NMRR](#), [MAIR0](#), [MAIR1](#), [AMAIR0](#), [AMAIR1](#), [CONTEXTIDR](#).

TRVM	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 read accesses to the specified Virtual Memory controls 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

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:
 - [SCTLR_EL1](#), [TTBR0_EL1](#), [TTBR1_EL1](#), [TCR_EL1](#), [ESR_EL1](#), [FAR_EL1](#), [AFSR0_EL1](#), [AFSR1_EL1](#), [MAIR_EL1](#), [AMAIR_EL1](#), [CONTEXTIDR_EL1](#).
- If EL1 is using AArch32 state, 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](#), [AMAIR0](#), [AMAIR1](#), [CONTEXTIDR](#).

TVM	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL1 write accesses to the specified EL1 virtual memory control 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.

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 RVALE1IS](#), [TLBI RVALE1IS](#), [TLBI RVALE1IS](#), [TLBI RVALE1IS](#).
 - If FEAT_TLBIOS and FEAT_TLBIRANGE are implemented, this trap applies to [TLBI RVAE1OS](#), [TLBI RVAE1OS](#), [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](#), [TLBIMVAAIS](#), [TLBIMVALIS](#), [TLBIMVAALIS](#).
 - [TLBIALL](#), [TLBIMVA](#), [TLBIASID](#), [TLBIMVAA](#), [TLBIMVAL](#), [TLBIMVAAL](#).
 - [ITLBIALL](#), [ITLBIIMVA](#), [ITLBIASID](#).
 - [DTLBIALL](#), [DTLBIIMVA](#), [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](#), [IC IALLU](#), [IC IALLUIS](#), [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.

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.

TCPC, bit [23]

When FEAT_DPB is implemented:

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 TCPC.

TCPC	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:

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 the value of HCR_EL2.TIDCP is 1, it is IMPLEMENTATION DEFINED whether any of this functionality accessed from EL0 is trapped to EL2. If it is not, then it is UNDEFINED, and any attempt to access it from EL0 generates an exception that is taken to EL1.

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

Arm expects the trapping of EL0 accesses to these functions to EL2 to be unusual, and used only when the hypervisor is virtualizing EL0 operation. Arm strongly recommends that unless the hypervisor must virtualize EL0 operation, an EL0 access to any of these functions is UNDEFINED, as it would be if the implementation did not include EL2. The PE then takes any resulting exception to EL1.

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](#).
 - 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.

When FEAT_WFxT or FEAT_WFxT2 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.

~~When FEAT_WFxT or FEAT_WFxT2 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 **bitfield** 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 **bitfield** 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
        return NVMem[0x078];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return HCR_EL2;
elsif PSTATE.EL == EL3 then
    return 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];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HCR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    HCR_EL2 = X[t];

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

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.

The reset behavior of this field is:

- 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, 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.

The reset behavior of this field is:

- 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.

The reset behavior of this field is:

- 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 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 SMPRMAP_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, 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 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 and EL2 is implemented, this field resets to 0.

Otherwise:

Reserved, RES0.

EnASR, bit [2]

When FEAT_LS64 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 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 and EL2 is implemented, this field resets to 0.

Otherwise:

Reserved, RES0.

EnAS0, bit [0]

When FEAT_LS64 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 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() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0xA0];
    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
        return HCRX_EL2;
elsif PSTATE.EL == EL3 then
    return 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];
    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];
elsif PSTATE.EL == EL3 then
    HCRX_EL2 = X[t];

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ee9a85f5e501ea

Copyright © 2010-2021 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** 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.

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 Aborts or Instruction Aborts 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 Aborts or Instruction Aborts 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:

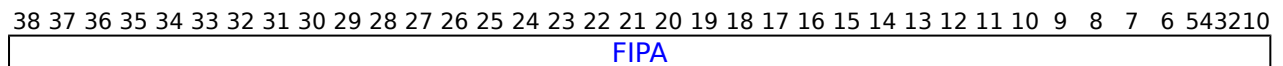
Reserved, RES0.

Bits [62:44]

Reserved, RES0.

FIPA, bits [43:4]

FIPA encoding when FEAT_LPA is implemented



FIPA, bits [38:0]

When 52-bit addresses are not in use for stage 1 translation, FIPA[38:35] is RES0.

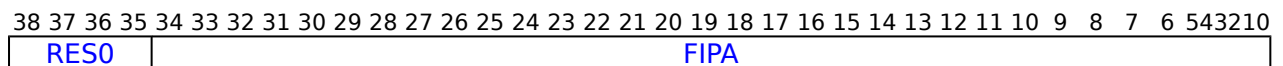
Bits [51:12] of the Faulting Intermediate Physical Address.

For implementations with 52-bit physical addresses, FIPA[38:35] forms the corresponding upper bits of this field. For implementations with fewer than 52 physical address bits, FIPA[38:35] is RES0.

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 [38:35]

Reserved, RES0.

FIPA, bits [34:0]

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.

The reset behavior of this field is:

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

(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

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'The instruction cache maintenance instruction (IC)'.

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, this instruction requires read access permission to the VA, otherwise it is IMPLEMENTATION DEFINED whether it generates a Permission fault, see 'Permission fault'.

When FEAT_CMOW is implemented, [HCR_EL2](#).{E2H, TGE} is not {1, 1}, [SCTLR_EL1](#).CMOW is 1, and EL0 is implemented, when executed at EL0, the instruction has stage 1 read permission to the VA, but does not have stage 1 write permission to the VA, the instruction generates a stage 1 Permission fault.

When FEAT_CMOW is implemented, [HCR_EL2](#).E2H is 1, [SCTLR_EL2](#).CMOW is 1, and EL0 access is enabled, when executed at EL0, the instruction has stage 1 read permission to the VA but does not have stage 1 write permission to the VA, the instruction generates a stage 1 Permission fault.

When FEAT_CMOW is implemented, [HCRX_EL2](#).CMOW is 1, and EL1 or EL0 access is enabled, when executed at EL1 or EL0 the instruction has stage 2 read permission to the VA but does not have stage 2 write permission to the VA, the instruction generates a stage 2 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') && 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.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' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HFGITR_EL2.ICIVAU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCI == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.IC(X[t], 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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGITR_EL2.ICIVAU == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.IC(X[t], CacheOpScope_PoU);
    elsif PSTATE.EL == EL2 then
        AArch64.IC(X[t], CacheOpScope_PoU);
    elsif PSTATE.EL == EL3 then
        AArch64.IC(X[t], CacheOpScope_PoU);

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

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] (S) are architecturally mapped to AArch32 System register [ICC_AP1R<n>\[31:0\]](#) (S).

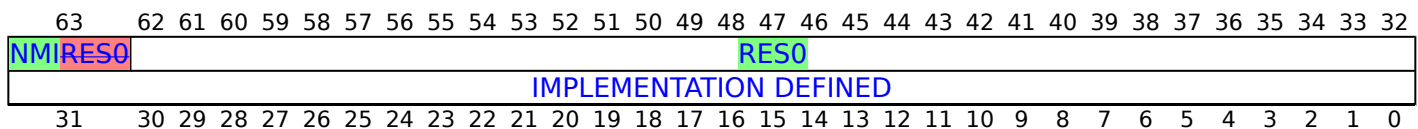
AArch64 System register ICC_AP1R<n>_EL1 bits [31:0] (NS) are architecturally mapped to AArch32 System register [ICC_AP1R<n>\[31:0\]](#) (NS).

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.

Field descriptions



NMI, Bits bit [63:32]

When FEAT_GICv3 NMI is implemented:

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<n>_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b0:n[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'" && 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
        return ICV_AP1R_EL1[UInt(op2<1:0>)];
    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
            return ICC_AP1R_EL1_S[UInt(op2<1:0>)];
        else
            return ICC_AP1R_EL1_NS[UInt(op2<1:0>)];
    else
        return ICC_AP1R_EL1[UInt(op2<1:0>)];
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
            return ICC_AP1R_EL1_S[UInt(op2<1:0>)];
        else
            return ICC_AP1R_EL1_NS[UInt(op2<1:0>)];
    else
        return ICC_AP1R_EL1[UInt(op2<1:0>)];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_AP1R_EL1_S[UInt(op2<1:0>)];
        else
            return ICC_AP1R_EL1_NS[UInt(op2<1:0>)];

```

MSR ICC_AP1R<n>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b0:n[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'" && 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_AP1R_EL1[UInt(op2<1:0>)] = X[t];
    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[UInt(op2<1:0>)] = X[t];
        else
            ICC_AP1R_EL1_NS[UInt(op2<1:0>)] = X[t];
    else
        ICC_AP1R_EL1[UInt(op2<1:0>)] = X[t];
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[UInt(op2<1:0>)] = X[t];
        else
            ICC_AP1R_EL1_NS[UInt(op2<1:0>)] = X[t];
    else
        ICC_AP1R_EL1[UInt(op2<1:0>)] = X[t];
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[UInt(op2<1:0>)] = X[t];
        else
            ICC_AP1R_EL1_NS[UInt(op2<1:0>)] = X[t];

```

3020/09/2021 14:12:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ICC_CTLR_EL3, Interrupt Controller Control Register (EL3)

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

AArch64 System register ICC_CTLR_EL3 bits [31:0] can be mapped to AArch32 System register [ICC_MCTLR\[31:0\]](#), but this is not architecturally mandated.

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

63	62	61	60	59	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												ExtRange		RSS	nDS	RES0	A3V	SEIS	IDbits	PRIbits	RES0	PMHE	RM	EOImode_EL1	NS	EOImode_EL1	NS	EOImode_EL1	NS						
31	30	29	28	27	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 [63:20]

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">Behaviour 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.

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 the Realm 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]

Possible values of this bit are:

Routing Modifier. This bit legacy controls operation whether of EL3EL1 can software acknowledge, or observe as the Highest Priority Pending Interrupt, Secure Group 0 and Non-secure Group 1 interrupts. with [GICC_CTLR.FIQEn](#) set to 1, this bit indicates whether interrupts can be acknowledged or observed as the Highest Priority Pending Interrupt, or whether a special INTID value is returned.

RM	Meaning
0b0	Secure Group 0 and Non-secure Group 1 interrupts can be acknowledged and observed as the highest priority interrupt at EL3. the Secure Exception level where the interrupt is taken.
0b1	Secure When Group accessed 0 at and EL3 Non-secure in Group 1 interrupts cannot be acknowledged and observed as the highest priority interrupt at EL3. state: <ul style="list-style-type: none"> • 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. 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
        return 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];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICC_IAR0_EL1, Interrupt Controller Interrupt Acknowledge Register 0

The ICC_IAR0_EL1 characteristics are:

Purpose

The PE reads this register to obtain the INTID of the signaled Group 0 interrupt. This read acts as an acknowledge for the interrupt.

Configuration

AArch64 System register ICC_IAR0_EL1 performs the same function as AArch32 System register [ICC_IAR0](#).

This register is present only when FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_IAR0_EL1 are UNDEFINED.

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when $PSTATE.\{I,F\} == \{0,0\}$). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. For more information, see 'Observability of the effects of accesses to the GIC registers'.

Attributes

ICC_IAR0_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								INTID																								
31	30	29	28	27	26	25	24	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.

INTID, bits [23:0]

The INTID of the signaled interrupt.

This is the INTID of the highest priority pending interrupt, if that interrupt is of sufficient priority for it to be signaled to the PE, and if it can be acknowledged at the current Security state and Exception level.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. **These special INTIDs can be one of: 1020, 1021, or 1023.** For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICC_CTLR_EL1.IDbits](#) and [ICC_CTLR_EL3.IDbits](#). If only 16 bits are implemented, bits [23:16] of this register are RES0.

Accessing ICC_IAR0_EL1

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

MRS <Xt>, ICC_IAR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	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'" && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FM0 == '1' then
        return ICV_IAR0_EL1;
    elsif HaveEL(EL3) && SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_IAR0_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
        return ICC_IAR0_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_IAR0_EL1;

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICC_IAR1_EL1, Interrupt Controller Interrupt Acknowledge Register 1

The ICC_IAR1_EL1 characteristics are:

Purpose

The PE reads this register to obtain the INTID of the signaled Group 1 interrupt. This read acts as an acknowledge for the interrupt.

Configuration

AArch64 System register ICC_IAR1_EL1 performs the same function as AArch32 System register [ICC_IAR1](#).

This register is present only when FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_IAR1_EL1 are UNDEFINED.

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when $PSTATE.\{I,F\} == \{0,0\}$). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. For more information, see 'Observability of the effects of accesses to the GIC registers' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Attributes

ICC_IAR1_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								INTID																								
31	30	29	28	27	26	25	24	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.

INTID, bits [23:0]

The INTID of the signaled interrupt.

This is the INTID of the highest priority pending interrupt, if that interrupt is of sufficient priority for it to be signaled to the PE, and if it can be acknowledged at the current Security state and Exception level.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. **These special INTIDs can be one of: 1020, 1021, or 1023.** For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICC_CTLR_EL1.IDbits](#) and [ICC_CTLR_EL3.IDbits](#). If only 16 bits are implemented, bits [23:16] of this register are RES0.

Accessing ICC_IAR1_EL1

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

MRS <Xt>, ICC_IAR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	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'" && 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
        return ICV_IAR1_EL1;
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_IAR1_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);
    else
        return ICC_IAR1_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_IAR1_EL1;

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

ICC_NMIAR1_EL1, Interrupt Controller Non-maskable Interrupt Acknowledge Register 1

The ICC_NMIAR1_EL1 characteristics are:

Purpose

The PE reads this register to obtain the INTID of the signaled Group 1 non-maskable interrupt. This read acts as an acknowledge for the interrupt.

Configuration

This register is present only when FEAT_GICv3_NMI is implemented. Otherwise, direct accesses to ICC_NMIAR1_EL1 are UNDEFINED.

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when $PSTATE.\{I,F\} == \{0,0\}$). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. For more information, see 'Observability of the effects of accesses to the GIC registers' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Attributes

ICC_NMIAR1_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																INTID															
31	30	29	28	27	26	25	24	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.

INTID, bits [23:0]

The INTID of the signaled interrupt.

This is the INTID of the highest priority pending interrupt, if that interrupt has the Non-maskable property and is of sufficient priority for it to be signalled to the PE, and if it can be acknowledged at the current Security state and Exception level.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICC_CTLR_EL1.IDbits](#) and [ICC_CTLR_EL3.IDbits](#). If only 16 bits are implemented, bits [23:16] of this register are RES0.

Accessing ICC_NMIAR1_EL1

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

MRS <Xt>, ICC_NMIAR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	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'" && 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
        return ICV_NMIAR1_EL1;
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_NMIAR1_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);
    else
        return ICC_NMIAR1_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_NMIAR1_EL1;

```

30/09/2021 14:52; 092b4e1bbfbb45a293b198f9330c5f529ead2b0f

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

The ICC_RPR_EL1 characteristics are:

Purpose

Indicates the Running priority of the CPU interface.

Configuration

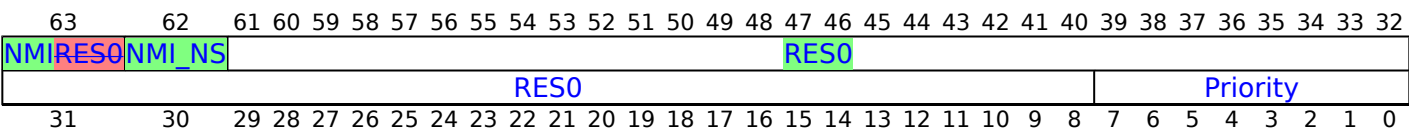
AArch64 System register ICC_RPR_EL1 performs the same function as AArch32 System register [ICC_RPR](#).

This register is present only when FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_RPR_EL1 are UNDEFINED.

Attributes

ICC_RPR_EL1 is a 64-bit register.

Field descriptions



NMI, Bits bit [63:8]
When FEAT_GICv3_NMI is implemented:

Indicates whether the running priority is from a NMI.

NMI	Meaning
0b0	<p>When <code>GICD_CTLR.DS=1</code>, there are no Active NMIs, or all Active NMIs have undergone priority drop.</p> <p>When <code>GICD_CTLR.DS=0</code>:</p> <ul style="list-style-type: none"> For Non-secure and Realm reads, there are no Active Non-secure Group 1 NMIs, or all Active Non-secure Group 1 NMIs have undergone priority drop. For Secure and Root reads, there are no Active Secure Group 1 NMIs, or all Active Secure Group 1 NMIs have undergone priority drop.
0b1	<p>When <code>GICD_CTLR.DS=1</code>, there is an Active NMI.</p> <p>When <code>GICD_CTLR.DS=0</code>:</p> <ul style="list-style-type: none"> For Non-secure and Realm reads, there is an Active Non-secure Group 1 NMIs. For Secure and Root reads, there is an Active Secure Group 1 NMIs.

Otherwise:

Reserved, RES0.

NMI_NS, bit [62]

When FEAT_GICv3_NMI is implemented and EL3 is implemented:

Indicates whether the running priority is from a Non-secure NMI.

NMI_NS	Meaning
0b0	There are no Active Non-secure Group 1 NMIs, or all Active Non-secure Group 1 NMIs have undergone priority drop.
0b1	There is an Active Non-secure Group 1 NMI which has not undergone priority drop.

Otherwise:

Reserved, RES0.

Bits [61:8]

Reserved, RES0.

Priority, bits [7:0]

The current running priority on the CPU interface. This is the group priority of the current active interrupt.

If there are no active interrupts on the CPU interface, or all active interrupts have undergone a priority drop, the value returned is the Idle priority.

The priority returned is the group priority as if the BPR for the current Exception level and Security state was set to the minimum value of BPR for the number of implemented priority bits.

Note

If 8 bits of priority are implemented the group priority is bits[7:1] of the priority.

Accessing ICC_RPR_EL1

Software cannot determine the number of implemented priority bits from a read of this register.

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

MRS <Xt>, ICC_RPR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	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'" && 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
        return ICV_RPR_EL1;
    elsif EL2Enabled() && HCR_EL2.IM0 == '1' then
        return ICV_RPR_EL1;
    elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return ICC_RPR_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);
            else
                return ICC_RPR_EL1;
    elsif PSTATE.EL == EL3 then
        if ICC_SRE_EL3.SRE == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return ICC_RPR_EL1;

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The ICH_AP1R<n>_EL2 characteristics are:

Purpose

Provides information about Group 1 virtual active priorities for EL2.

Configuration

AArch64 System register ICH_AP1R<n>_EL2 bits [31:0] are architecturally mapped to AArch32 System register [ICH_AP1R<n>\[31:0\]](#).

This register is present only when FEAT_GICv3 is implemented and (EL2 is implemented or EL3 is implemented). Otherwise, direct accesses to ICH AP1R<n> 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

ICH AP1R<n> 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		
NMIRESO														RESO																		
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		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		

NMI, Bits bit [63:32]

When FEAT_GICv3_NMI is implemented:

Indicates whether the running virtual priority is from a NMI.

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.

P<x>, bit [x], for x = 31 to 0

Group 1 interrupt active priorities. Possible values of each bit are:

P<x>	Meaning
0b0	There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop.
0b1	There is a Group 1 interrupt active with this priority level which has not undergone priority drop.

The correspondence between priority levels and bits depends on the number of bits of priority that are implemented.

If 5 bits of preemption are implemented (bits [7:3] of priority), then there are 32 preemption levels, and the active state of these preemption levels are held in ICH_AP1R0_EL2 in the bits corresponding to Priority[7:3].

If 6 bits of preemption are implemented (bits [7:2] of priority), then there are 64 preemption levels, and:

- The active state of preemption levels 0 - 124 are held in ICH_AP1R0_EL2 in the bits corresponding to 0:Priority[6:2].
- The active state of preemption levels 128 - 252 are held in ICH_AP1R1_EL2 in the bits corresponding to 1:Priority[6:2].

If 7 bits of preemption are implemented (bits [7:1] of priority), then there are 128 preemption levels, and:

- The active state of preemption levels 0 - 62 are held in ICH_AP1R0_EL2 in the bits corresponding to 00:Priority[5:1].
- The active state of preemption levels 64 - 126 are held in ICH_AP1R1_EL2 in the bits corresponding to 01:Priority[5:1].
- The active state of preemption levels 128 - 190 are held in ICH_AP1R2_EL2 in the bits corresponding to 10:Priority[5:1].
- The active state of preemption levels 192 - 254 are held in ICH_AP1R3_EL2 in the bits corresponding to 11:Priority[5:1].

Note

Having the bit corresponding to a priority set to 1 in both [ICH_AP0R<n>_EL2](#) and ICH_AP1R<n>_EL2 might result in UNPREDICTABLE behavior of the interrupt prioritization system for virtual interrupts.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

This register is always used for legacy VMs, regardless of the group of the virtual interrupt. Reads and writes to [GICV_APR<n>](#) access [ICH_AP1R<n>_EL2](#). For more information about support for legacy VMs, see 'Support for legacy operation of VMs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Accessing ICH_AP1R<n>_EL2

ICH_AP1R1_EL2 is only implemented in implementations that support 6 or more bits of preemption. ICH_AP1R2_EL2 and ICH_AP1R3_EL2 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_EL2](#).PREbits

Writing to these registers with any value other than the last read value of the register (or 0x00000000 for a newly set up virtual machine) can result in UNPREDICTABLE behavior of the virtual interrupt prioritization system allowing either:

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

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

MRS <Xt>, ICH_AP1R<n>_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	0b0:n[1:0]

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x4A0+8*UInt(op2<1:0>)];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ICH_AP1R_EL2[UInt(op2<1:0>)];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICH_AP1R_EL2[UInt(op2<1:0>)];

```

MSR ICH_AP1R<n>_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b1001	0b0:n[1:0]

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x4A0+8*UInt(op2<1:0>)] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_AP1R_EL2[UInt(op2<1:0>)] = X[t];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICH_AP1R_EL2[UInt(op2<1:0>)] = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICH_LR<n>_EL2, Interrupt Controller List Registers, n = 0 - 15

The ICH_LR<n>_EL2 characteristics are:

Purpose

Provides interrupt context information for the virtual CPU interface.

Configuration

AArch64 System register ICH_LR<n>_EL2 bits [31:0] are architecturally mapped to AArch32 System register [ICH_LR<n>\[31:0\]](#).

AArch64 System register ICH_LR<n>_EL2 bits [63:32] are architecturally mapped to AArch32 System register [ICH_LRC<n>\[31:0\]](#).

This register is present only when FEAT_GICv3 is implemented and (EL2 is implemented or EL3 is implemented). Otherwise, direct accesses to ICH_LR<n>_EL2 are UNDEFINED.

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

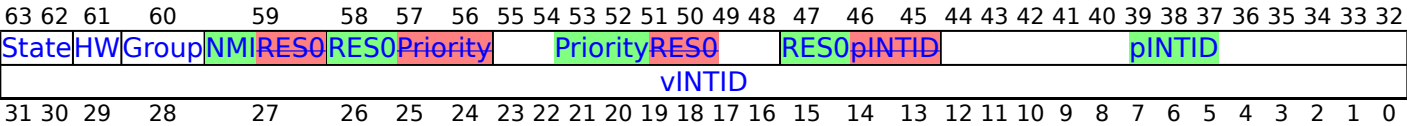
If list register n is not implemented, then accesses to this register are UNDEFINED.

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

Attributes

ICH_LR<n>_EL2 is a 64-bit register.

Field descriptions



State, bits [63:62]

The state of the interrupt:

State	Meaning
0b00	Invalid (Inactive).
0b01	Pending.
0b10	Active.
0b11	Pending and active.

The GIC updates these state bits as virtual interrupts proceed through the interrupt life cycle. Entries in the invalid state are ignored, except for the purpose of generating virtual maintenance interrupts.

For hardware interrupts, the pending and active state is held in the physical Distributor rather than the virtual CPU interface. A hypervisor must only use the pending and active state for software originated interrupts, which are typically associated with virtual devices, or SGIs.

The reset behavior of this field is:

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

HW, bit [61]

Indicates whether this virtual interrupt maps directly to a hardware interrupt, meaning that it corresponds to a physical interrupt. Deactivation of the virtual interrupt also causes the deactivation of the physical interrupt with the ID that the pINTID field indicates.

HW	Meaning
0b0	The interrupt is triggered entirely by software. No notification is sent to the Distributor when the virtual interrupt is deactivated.
0b1	The interrupt maps directly to a hardware interrupt. A deactivate interrupt request is sent to the Distributor when the virtual interrupt is deactivated, using the pINTID field from this register to indicate the physical interrupt ID. If ICH_VMCR_EL2.VEOIM is 0, this request corresponds to a write to ICC_EOIR0_EL1 or ICC_EOIR1_EL1 . Otherwise, it corresponds to a write to ICC_DIR_EL1 .

The reset behavior of this field is:

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

Group, bit [60]

Indicates the group for this virtual interrupt.

Group	Meaning
0b0	This is a Group 0 virtual interrupt. ICH_VMCR_EL2.VFIQEn determines whether it is signaled as a virtual IRQ or as a virtual FIQ, and ICH_VMCR_EL2.VENG0 enables signaling of this interrupt to the virtual machine.
0b1	This is a Group 1 virtual interrupt, signaled as a virtual IRQ. ICH_VMCR_EL2.VENG1 enables the signalling of this interrupt to the virtual machine. If ICH_VMCR_EL2.VCBPR is 0, then ICC_BPR1_EL1 determines if a pending Group 1 interrupt has sufficient priority to preempt current execution. Otherwise, ICH_LR<n>_EL2 determines preemption.

The reset behavior of this field is:

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

NMI, Bits bit [59:56]

When [FEAT_GICv3_NMI](#) is implemented:

Indicates whether the virtual priority has the non-maskable property.

NMI	Meaning
0b0	vINTID does not have the non-maskable interrupt property.
0b1	vINTID has the non-maskable interrupt property.

Setting [ICH_LR<n>_EL2.NMI](#) to 1 when [ICH_LR<n>_EL2.State](#) is not Invalid is CONSTRAINED UNPREDICTABLE if either [ICH_LR<n>_EL2.vINTID](#) indicates an LPI or [ICH_LR<n>_EL2.Group](#) is 0.

The permitted behaviours are:

- [ICH_LR<n>_EL2.NMI](#) is treated as 0 for all purposes other than a direct read of the register.
- The virtual interrupt is presented with superpriority.

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.

Priority, bits [55:48]

The priority of this interrupt.

It is IMPLEMENTATION DEFINED how many bits of priority are implemented, though at least five bits must be implemented. Unimplemented bits are RES0 and start from bit[48] up to bit[50]. The number of implemented bits can be discovered from [ICH_VTR_EL2.PRIBits](#).

When ICH_LR<n>_EL2.NMI is set to 1, this field is RES0 and the virtual interrupt's priority is treated as 0x00.

The reset behavior of this field is:

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

Bits [47:45]

Reserved, RES0.

pINTID, bits [44:32]

Physical INTID, for hardware interrupts.

When ICH_LR<n>_EL2.HW is 0 (there is no corresponding physical interrupt), this field has the following meaning:

- Bits[44:42] : RES0.
- Bit[41] : EOI. If this bit is 1, then when the interrupt identified by vINTID is deactivated, a maintenance interrupt is asserted.
- Bits[40:32] : RES0.

When ICH_LR<n>_EL2.HW is 1 (there is a corresponding physical interrupt):

- This field indicates the physical INTID. This field is only required to implement enough bits to hold a valid value for the implemented INTID size. Any unused higher order bits are RES0.
- When [ICC_CTLR_EL1.ExtRange](#) is 0, then bits[44:42] of this field are RES0.
- If the value of pINTID is not a valid INTID, behavior is UNPREDICTABLE. If the value of pINTID indicates a PPI, this field applies to the PPI associated with this same physical PE ID as the virtual CPU interface requesting the deactivation.

A hardware physical identifier is only required in List Registers for interrupts that require deactivation. This means only 13 bits of Physical INTID are required, regardless of the number specified by [ICC_CTLR_EL1.IDbits](#).

The reset behavior of this field is:

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

vINTID, bits [31:0]

Virtual INTID of the interrupt.

If the value of vINTID is 1020-1023 and ICH_LR<n>_EL2.State!=0b00 (Inactive), behavior is UNPREDICTABLE.

Behavior is UNPREDICTABLE if two or more List Registers specify the same vINTID when:

- ICH_LR<n>_EL2.State == 0b01.
- ICH_LR<n>_EL2.State == 0b10.
- ICH_LR<n>_EL2.State == 0b11.

It is IMPLEMENTATION DEFINED how many bits are implemented, though at least 16 bits must be implemented. Unimplemented bits are RES0. The number of implemented bits can be discovered from [ICH_VTR_EL2.IDbits](#).

When [ICC_SRE_EL1.SRE](#) == 0, specifying a vINTID in the LPI range is UNPREDICTABLE

Note

When a VM is using memory-mapped access to the GIC, software must ensure that the correct source PE ID is provided in bits[12:10].

The reset behavior of this field is:

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

Accessing ICH_LR<n>_EL2

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

MRS <Xt>, ICH_LR<n>_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b110:n[3]	n[2:0]

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x400+8*UInt(CRm<0>:op2<2:0>)];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ICH_LR_EL2[UInt(CRm<0>:op2<2:0>)];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICH_LR_EL2[UInt(CRm<0>:op2<2:0>)];

```

MSR ICH_LR<n>_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1100	0b110:n[3]	n[2:0]

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x400+8*UInt(CRm<0>:op2<2:0>)] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        ICH_LR_EL2[UInt(CRm<0>:op2<2:0>)] = X[t];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ICH_LR_EL2[UInt(CRm<0>:op2<2:0>)] = X[t];

```

3020/09/2021 1412:5237: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The ICV_AP1R<n>_EL1 characteristics are:

Purpose

Provides information about virtual Group 1 active priorities.

Configuration

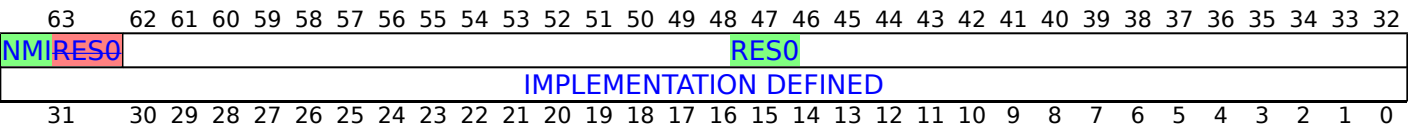
AArch64 System register ICV_AP1R<n>_EL1 bits [31:0] are architecturally mapped to AArch32 System register ICV_AP1R<n>[31:0].

This register is present only when FEAT_GICv3 is implemented and EL2 is implemented. Otherwise, direct accesses to ICV AP1R<n>_EL1 are UNDEFINED.

Attributes

ICV_AP1R<n>_EL1 is a 64-bit register.

Field descriptions



NMI, Bits bit [63:32]
When FEAT_GICv3_NMI is implemented:

Indicates whether the running priority is from a NMI.

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 an architecturally UNKNOWN value.

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 ICV_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 virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV_AP1R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICV_AP1R2_EL1 and ICV_AP1R3_EL1 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

Writing to the active priority registers in any order other than the following order might result in UNPREDICTABLE behavior of the interrupt prioritization system:

- [ICV_AP0R<n>_EL1](#).
- ICV_AP1R<n>_EL1.

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

MRS <Xt>, ICC_AP1R<n>_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b0:n[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'" && 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
        return ICV_AP1R_EL1[UInt(op2<1:0>)];
    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
            return ICC_AP1R_EL1_S[UInt(op2<1:0>)];
        else
            return ICC_AP1R_EL1_NS[UInt(op2<1:0>)];
    else
        return ICC_AP1R_EL1[UInt(op2<1:0>)];
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
            return ICC_AP1R_EL1_S[UInt(op2<1:0>)];
        else
            return ICC_AP1R_EL1_NS[UInt(op2<1:0>)];
    else
        return ICC_AP1R_EL1[UInt(op2<1:0>)];
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_AP1R_EL1_S[UInt(op2<1:0>)];
        else
            return ICC_AP1R_EL1_NS[UInt(op2<1:0>)];

```

MSR ICC_AP1R<n>_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b0:n[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'" && 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_AP1R_EL1[UInt(op2<1:0>)] = X[t];
    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[UInt(op2<1:0>)] = X[t];
        else
            ICC_AP1R_EL1_NS[UInt(op2<1:0>)] = X[t];
    else
        ICC_AP1R_EL1[UInt(op2<1:0>)] = X[t];
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[UInt(op2<1:0>)] = X[t];
        else
            ICC_AP1R_EL1_NS[UInt(op2<1:0>)] = X[t];
    else
        ICC_AP1R_EL1[UInt(op2<1:0>)] = X[t];
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[UInt(op2<1:0>)] = X[t];
        else
            ICC_AP1R_EL1_NS[UInt(op2<1:0>)] = X[t];

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICV_IAR0_EL1, Interrupt Controller Virtual Interrupt Acknowledge Register 0

The ICV_IAR0_EL1 characteristics are:

Purpose

The PE reads this register to obtain the INTID of the signaled virtual Group 0 interrupt. This read acts as an acknowledge for the interrupt.

Configuration

AArch64 System register ICV_IAR0_EL1 performs the same function as AArch32 System register [ICV_IAR0](#).

This register is present only when FEAT_GICv3 is implemented and EL2 is implemented. Otherwise, direct accesses to ICV_IAR0_EL1 are UNDEFINED.

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when $PSTATE.\{I,F\} == \{0,0\}$). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. For more information, see 'Observability of the effects of accesses to the GIC registers' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Attributes

ICV_IAR0_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								INTID																								
31	30	29	28	27	26	25	24	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.

INTID, bits [23:0]

The INTID of the signaled virtual interrupt.

This is the INTID of the highest priority pending virtual interrupt, if that interrupt is of sufficient priority for it to be signaled to the PE, and if it can be acknowledged.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. **This special INTID can take the value 1023 only.** For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICV_CTLR_EL1.IDbits](#). If only 16 bits are implemented, bits [23:16] of this register are RES0.

Accessing ICV_IAR0_EL1

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

MRS <Xt>, ICC_IAR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	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'" && SCR_EL3.FIQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FM0 == '1' then
        return ICV_IAR0_EL1;
    elsif HaveEL(EL3) && SCR_EL3.FIQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_IAR0_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
        return ICC_IAR0_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_IAR0_EL1;

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICV_IAR1_EL1, Interrupt Controller Virtual Interrupt Acknowledge Register 1

The ICV_IAR1_EL1 characteristics are:

Purpose

The PE reads this register to obtain the INTID of the signaled virtual Group 1 interrupt. This read acts as an acknowledge for the interrupt.

Configuration

AArch64 System register ICV_IAR1_EL1 performs the same function as AArch32 System register [ICV_IAR1](#).

This register is present only when FEAT_GICv3 is implemented and EL2 is implemented. Otherwise, direct accesses to ICV_IAR1_EL1 are UNDEFINED.

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when $PSTATE.\{I,F\} == \{0,0\}$). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. For more information, see 'Observability of the effects of accesses to the GIC registers' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Attributes

ICV_IAR1_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								INTID																								
31	30	29	28	27	26	25	24	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.

INTID, bits [23:0]

The INTID of the signaled virtual interrupt.

This is the INTID of the highest priority pending virtual interrupt, if that interrupt is of sufficient priority for it to be signaled to the PE, and if it can be acknowledged.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. **This special INTID can take the value 1023 only.** For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICV_CTLR_EL1.IDbits](#). If only 16 bits are implemented, bits [23:16] of this register are RES0.

Accessing ICV_IAR1_EL1

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

MRS <Xt>, ICC_IAR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	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'" && 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
        return ICV_IAR1_EL1;
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_IAR1_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);
    else
        return ICC_IAR1_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_IAR1_EL1;

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

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

The ICV_NMIAR1_EL1 characteristics are:

Purpose

The PE reads this register to obtain the INTID of the signaled virtual Group 1 interrupt. This read acts as an acknowledge for the interrupt.

Configuration

This register is present only when FEAT_GICv3_NMI is implemented and EL2 is implemented. Otherwise, direct accesses to ICV_NMIAR1_EL1 are UNDEFINED.

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when $PSTATE.\{I,F\} == \{0,0\}$). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. For more information, see 'Observability of the effects of accesses to the GIC registers' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Attributes

ICV_NMIAR1_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																INTID															
31	30	29	28	27	26	25	24	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.

INTID, bits [23:0]

The INTID of the signaled virtual interrupt.

This is the INTID of the highest priority pending virtual interrupt, if that virtual interrupt has the Non-maskable property and is of sufficient priority for it to be signalled to the PE, and if it can be acknowledged at the current Security state and Exception level.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. For more information, see 'Special INTIDs' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICV_CTLR_EL1.IDbits](#). If only 16 bits are implemented, bits [23:16] of this register are RES0.

Accessing ICV_NMIAR1_EL1

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

MRS <Xt>, ICC_NMIAR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	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'" && 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
        return ICV_NMIAR1_EL1;
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_NMIAR1_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);
    else
        return ICC_NMIAR1_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_NMIAR1_EL1;

```

30/09/2021 14:52; 092b4e1bbfbb45a293b198f9330c5f529ead2b0f

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

The ICV_RPR_EL1 characteristics are:

Purpose

Indicates the Running priority of the virtual CPU interface.

Configuration

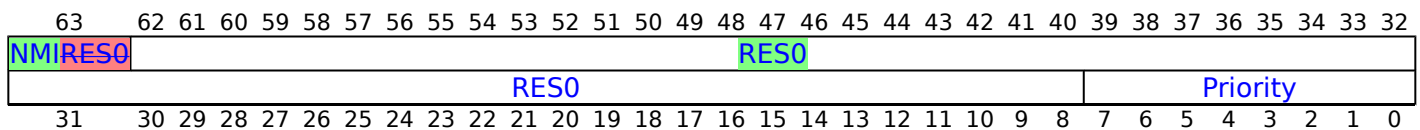
AArch64 System register ICV_RPR_EL1 performs the same function as AArch32 System register [ICV_RPR](#).

This register is present only when FEAT_GICv3 is implemented and EL2 is implemented. Otherwise, direct accesses to ICV_RPR_EL1 are UNDEFINED.

Attributes

ICV_RPR_EL1 is a 64-bit register.

Field descriptions



NMI, Bits bit [63:8]

When FEAT_GICv3_NMI is implemented:

Indicates whether the running priority is from a NMI.

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.

Otherwise:

Reserved, RES0.

Bits [62:8]

Reserved, RES0.

Priority, bits [7:0]

The current running priority on the virtual CPU interface. This is the group priority of the current active virtual interrupt.

If there are no active interrupts on the virtual CPU interface, or all active interrupts have undergone a priority drop, the value returned is the Idle priority.

The priority returned is the group priority as if the BPR for the current Exception level and Security state was set to the minimum value of BPR for the number of implemented priority bits.

Note

If 8 bits of priority are implemented the group priority is bits[7:1] of the priority.

Accessing ICV_RPR_EL1

If there are no active interrupts on the virtual CPU interface, or all active interrupts have undergone a priority drop, the value returned is the Idle priority.

Software cannot determine the number of implemented priority bits from a read of this register.

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

MRS <Xt>, ICC_RPR_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1011	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'" && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elseif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.FM0 == '1' then
        return ICV_RPR_EL1;
    elseif EL2Enabled() && HCR_EL2.IM0 == '1' then
        return ICV_RPR_EL1;
    elseif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_RPR_EL1;
elseif 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;
    elseif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_RPR_EL1;
elseif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ICC_RPR_EL1;

```

3020/09/2021 14:12:53; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

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			
HPMN0RES0				RES0BRBE				BRBEMTPMU				MTPMUTraceBuffer				TraceBuffer				TraceFilter				TraceFilterDoubleLock				DoubleLockPMSVer			
CTX CMPs				RES0				WRPs				RES0				BRPs				PMUVer				TraceVer							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3			

HPMN0, Bits bits [63:6056]

Zero PMU event counters for a Guest operating system. Defined values are:

HPMN0	Meaning
0b0000	Setting <code>MDCR_EL2.HPMN</code> to zero has CONstrained UNPREDICTABLE behavior.
0b0001	Setting <code>MDCR_EL2.HPMN</code> 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.

Bits [59:56]

Reserved, RES0.

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 , implemented , FEAT_BRBE .
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, FEAT_TRBE.

All other values are reserved.

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 PE 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 Event 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.

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.

In Armv8.5, if FEAT_SPE is implemented, the value 0b0001 is not permitted.

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.

CTX_CMPs, bits [31:28]

Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints.

Bits [27:24]

Reserved, RES0.

WRPs, bits [23:20]

Number of watchpoints, minus 1. The value of 0b0000 is reserved.

Bits [19:16]

Reserved, RES0.

BRPs, bits [15:12]

Number of breakpoints, minus 1. 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 also includes 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. control bit.
0b0101	PMUv3 for Armv8.4. As 0b0100, and adds also includes support for the PMMIR_EL1 register.
0b0110	PMUv3 for Armv8.5. As 0b0101, and adds also includes support for: <ul style="list-style-type: none"> 64-bit event counters. If EL2 is implemented, the MDCR_EL2.HCCD control. control bit. If EL3 is implemented, the MDCR_EL3.SCCD control. control bit.
0b0111	PMUv3 for Armv8.7. As 0b0110, and adds also includes support for: <ul style="list-style-type: none"> The PMCR_EL0.FZO and, if EL2 is implemented, MDCR_EL2.HPMFZO controls. control bits. If EL3 is implemented, the MDCR_EL3.{MPMX,MCCD} controls. control bits.
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.
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.

In Armv8.1, if FEAT_PMUv3 is implemented, the value 0b0001 is not permitted.

From In Armv8.1 Armv8.4, if FEAT_PMUv3 is implemented, the value 0b00010b0100 is not permitted.

From In Armv8.4 Armv8.5, if FEAT_PMUv3 is implemented, the value 0b01000b0101 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.

TraceVer, bits [7:4]

Trace support. Indicates whether System register interface to a PE trace unit is implemented. Defined values are:

TraceVer	Meaning
0b0000	PE trace unit System registers not implemented.
0b0001	PE trace unit System registers implemented.

All other values are reserved.

When PE 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, architecture. FEAT_Debugv8p2.
0b1001	Armv8.4 debug architecture, architecture. FEAT_Debugv8p4.
0b1010	Armv8.8 debug architecture, FEAT_Debugv8p8.

All other values are reserved.

FEAT_VHEFEAT_Debugv8p2 adds the functionality identified by the value 0b01110b1000.

FEAT_Debugv8p2FEAT_Debugv8p4 adds the functionality identified by the value 0b10000b1001.

FEAT_Debugv8p4 addsIn the functionality identified byArmv8.1, the value 0b10010b0110 is not permitted.

FEAT_Debugv8p8 addsIn the functionality identified byArmv8.2, the value 0b10100b0111 is not permitted.

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.

(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

- The [TCR_EL1](#).{TBID,TBID0}, [TCR_EL2](#).{TBID0,TBID1}, [TCR_EL2](#).TBID and [TCR_EL3](#).TBID bits are RES0.
- [APIAKeyHi_EL1](#), [APIAKeyLo_EL1](#), [APIBKeyHi_EL1](#), [APIBKeyLo_EL1](#), [APDAKeyHi_EL1](#), [APDAKeyLo_EL1](#), [APDBKeyHi_EL1](#), [APDBKeyLo_EL1](#) are not allocated.
- [SCTLR_ELx.EnIA](#), [SCTLR_ELx.EnIB](#), [SCTLR_ELx.EnDA](#), [SCTLR_ELx.EnDB](#) are all RES0.

If ID_AA64ISAR1_EL1.{GPI, GPA, API, APA} == {0000, 0000, 0000, 0000}, then:

- [HCR_EL2](#).APK and [HCR_EL2](#).API are RES0.
- [SCR_EL3](#).APK and [SCR_EL3](#).API are RES0.

There are ID_AA64ISAR1_EL1.{API, noAPA} configuration == notes.{0000, 0000}, then:

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 and ST64B* 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 LD64 and ST64B* 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 EPCR .EBF field is also supported.

All other values are reserved.

FEAT_BF16 implements the functionality identified by 0b0001.

FEAT_EBF16 implements the functionality identified by 0b0010.

When Advanced SIMD and SVE are both implemented, this field must return the same value as [ID_AA64ZFR0_EL1](#).BF16.

From Armv8.6, if FEAT_SME is implemented, the permitted values are 0b0001 and 0b0010.

From Armv8.6, if FEAT_SME is not implemented, the only permitted value is 0b0001.

SPECRES, bits [43:40]

Indicates support for prediction invalidation instructions in AArch64 state. Defined values are:

SPECRES	Meaning
0b0000	CFP RCTX, DVP RCTX, and CPP RCTX instructions are not implemented.
0b0001	CFP RCTX, DVP RCTX, and CPP RCTX instructions are implemented.

All other values are reserved.

FEAT_SPECRES 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.

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 this field must have~~ 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 this field must have~~ 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	The LDAPR*, LDAPUR*, and STLUR* instructions are not implemented.
0b0001	The LDAPR* instructions are implemented. The LDAPUR*, and STLUR* instructions are not implemented.
0b0010	The LDAPR*, LDAPUR*, and STLUR* instructions are implemented.

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.

In Armv8.2, the permitted values are 0b0000, 0b0001, and 0b0010.

In Armv8.3, the permitted values are 0b0001 and 0b0010.

From Armv8.4, the only permitted value is 0b0010.

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_PAuth implements the functionality added by the values 0b0000, 0b0001, and 0b0010.~~

~~FEAT_EPACFEAT_PAuth2~~ implements the functionality ~~identified~~added by ~~the value~~ 0b0010~~0b0011~~.

FEAT_PAuth2 implements the functionality identified by 0b0011.

~~FEAT_FPAC implements the functionality added by the values 0b0100 and 0b0101.~~

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 this field must have~~ 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_PAuth implements the functionality added by the values 0b0000, 0b0001, and 0b0010.~~

~~FEAT_EPACFEAT_PAuth2~~ implements the functionality ~~identified~~added by ~~the value~~ 0b0010~~0b0011~~.

FEAT_PAuth2 implements the functionality identified by 0b0011.

~~FEAT_FPAC implements the functionality added by the values 0b0100 and 0b0101.~~

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 this field must have~~ 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.

Accessing ID_AA64ISAR1_EL1

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

MRS <Xt>, ID_AA64ISAR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	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
        return ID_AA64ISAR1_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64ISAR1_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64ISAR1_EL1;

```

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

(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																																			
RES0				PAC_frac				RPRES				BCWfxT				MOPS				APA3				GPA3				RPRES				WfxT			
31	30	29	28	27	26	25	24	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.

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]

When FPCR.AH is 1, indicates support for 12 bits of mantissa in reciprocal and reciprocal square root instructions in AArch64 state. Defined values are:

RPRES	Meaning
0b0000	Reciprocal and reciprocal square root estimates give 8 bits of mantissa.
0b0001	Reciprocal and reciprocal square root estimates give 12 bits of mantissa.

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.
0b0001	WFET and WFIT are supported, but the register number is not reported in the ESR_ELx on exceptions.
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 0b00100b0001.

From Armv8.7, the permitted values are 0b0001 and 0b0010.

Note

~~FEAT_WFxT2 implements the functionality identified by the value 0b0010.~~

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

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0110	0b010

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdfb36e47856c443a7ce9a85f5e501ca

(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
RES0				CMOWnTLBPA				TIDCP1AFP				nTLBPAHCX				AFPETS				HCXTWED				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

Bits [63:6052]

Reserved, RES0.

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 caches of previous valid translation table entries since the last completed relevant TLBI applicable to the PE where either: <ul style="list-style-type: none"> 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.
0b0001	The intermediate caching of translation table walks does not include non-coherent caches of previous valid translation table entries since the last completed TLBI applicable to the PE where either: <ul style="list-style-type: none"> 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_ETC 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]

Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches. The defined values of this field are:

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.

PAN, bits [23:20]

Privileged Access Never. Indicates support for the PAN bit in PSTATE, [SPSR_EL1](#), [SPSR_EL2](#), [SPSR_EL3](#), and [DPSR_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
        return ID_AA64MMFR1_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64MMFR1_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64MMFR1_EL1;

```

3020/09/2021 1412:5237: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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_BBIM 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
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.

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() && (!IsZero(ID_AA64MMFR2_EL1) || boolean IMPLEMENTATION_DEFINED
"ID_AA64MMFR2_EL1ID_AA64MMFR2 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            return ID_AA64MMFR2_EL1;
    elsif PSTATE.EL == EL2 then
        return ID_AA64MMFR2_EL1;
    elsif PSTATE.EL == EL3 then
        return ID_AA64MMFR2_EL1;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

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				EL0			
31	30	29	28	27	26	25	24	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, or 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_EOPD 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	This PE does not disclose whether branch targets trained in one hardware-described context can exploitatively control speculative execution in a different hardware-described context.
0b0001	Branch targets trained in one hardware-described context can exploitatively control speculative execution in a different hardware-described context only in a hard-to-determine way. Contexts do not include the SCXTNUM_ELx register contexts. Support for the SCXTNUM_ELx registers is defined in ID_AA64PFR1_EL1.CSV2_frac .
0b0010	Branch targets trained in one hardware-described context can exploitatively control speculative execution in a different hardware-described context only in a hard-to-determine way. The SCXTNUM_ELx registers are supported and the contexts include the SCXTNUM_ELx register contexts.

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.

In Armv8.0, the permitted values are 0b0000, 0b0001, and 0b0010.

From Armv8.5, the permitted values are 0b0001 and 0b0010.

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 support for MPAM Extension. Defined values are:

MPAM	Meaning
0b0000	If ID_AA64PFR1_EL1 .MPAM_frac == 0b0000, MPAM Extension is not implemented. If ID_AA64PFR1_EL1 .MPAM_frac == 0b0001, MPAM Extension version 0.1 is implemented.
0b0001	If ID_AA64PFR1_EL1 .MPAM_frac == 0b0000, MPAM Extension version 1.0 is implemented. If ID_AA64PFR1_EL1 .MPAM_frac == 0b0001, MPAM Extension version 1.1 is implemented.

All other values are reserved.

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.

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, ERXPFGCTL_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_RAS implements the functionality identified by the value 0b0001.

FEAT_RASv1p1 and FEAT_DoubleFault implement the functionality identified by the value 0b0010.

In Armv8.0 and Armv8.1, the permitted values are 0b0000 and 0b0001.

In Armv8.2, the only permitted value is 0b0001.

From Armv8.4, if FEAT_DoubleFault is implemented, the only permitted value is 0b0010.

From Armv8.4, when FEAT_DoubleFault is not implemented, and [ERRIDR_EL1](#) is 0, the permitted values are IMPLEMENTATION DEFINED 0b0001 or 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;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TID3 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            return ID_AA64PFR0_EL1;
    elseif PSTATE.EL == EL2 then
        return ID_AA64PFR0_EL1;
    elseif PSTATE.EL == EL3 then
        return ID_AA64PFR0_EL1;

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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
RES0																								NMICSV2_frac		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

Bits [63:4036]

Reserved, RES0.

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	This PE does not disclose whether branch targets trained in one hardware-described context can exploitatively control speculative execution in a different hardware-described context. The SCXTNUM_ELx registers are not supported.
0b0001	If ID_AA64PFR0_EL1.CSV2 is 0b0001, branch targets trained in one hardware-described context can exploitatively control speculative execution in a different hardware-described context only in a hard-to-determine way. Within a hardware-described context, branch targets trained for branches situated at one address can control speculative execution of branches situated at different addresses only in a hard-to-determine way. The SCXTNUM_ELx registers are not supported and the contexts do not include the SCXTNUM_ELx register contexts.
0b0010	If ID_AA64PFR0_EL1.CSV2 is 0b0001, branch targets trained in one hardware-described context can exploitatively control speculative execution in a different hardware-described context only in a hard-to-determine way. Within a hardware-described context, branch targets trained for branches situated at one address can control speculative execution of branches situated at different addresses only in a hard-to-determine way. The SCXTNUM_ELx registers are supported, but the contexts do not include the SCXTNUM_ELx register contexts.

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.

This field is valid only if [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 field. Defined values are:

SME	Meaning
0b0000	SME architectural state and programmers' model are not implemented.
0b0001	SME architectural state and programmers' model are implemented.

All other values are reserved.

FEAT_SME implements the functionality identified by the value 0b0001.

From Armv9.2, the permitted values are 0b0000 and 0b0001.

Bits [23:20]

Reserved, RES0.

MPAM_frac, bits [19:16]

MPAM Extension fractional field. Defined values are:

MPAM_frac	Meaning
0b0000	If ID_AA64PFR0_EL1 .MPAM == 0b0000, MPAM Extension not implemented.
0b0001	If ID_AA64PFR0_EL1 .MPAM == 0b0001, MPAM Extension v1.0 is implemented.
0b0001	If ID_AA64PFR0_EL1 .MPAM == 0b0000, implements MPAM v0.1, which is like v1.1 but reduces support for Secure PARTIDs.
0b0001	If ID_AA64PFR0_EL1 .MPAM == 0b0001, implements MPAM v1.1 and adds support for MPAM2_EL2 .TIDR to provide trapping of MPAMIDR_EL1 when MPAMHCR_EL2 is not present.

All other values are reserved.

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.
0b0011	Memory Tagging Extension is implemented with support for asymmetric Tag Check Fault handling.

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 and 0b0010.

From Armv8.7, the value 0b0001 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 AArch64 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypassing Safe, and the MSR and MRS instructions to directly read and write the PSTATE.SSBS field, 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
            return ID_AA64PFR1_EL1;
    elseif PSTATE.EL == EL2 then
        return ID_AA64PFR1_EL1;
    elseif PSTATE.EL == EL3 then
        return ID_AA64PFR1_EL1;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ID_AA64SMFR0_EL1, SME Feature ID register 0

The ID_AA64SMFR0_EL1 characteristics are:

Purpose

Provides information about the implemented features of the AArch64 Scalable Matrix Extension, when the [ID_AA64PFR1_EL1](#).SME field is not zero.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configuration

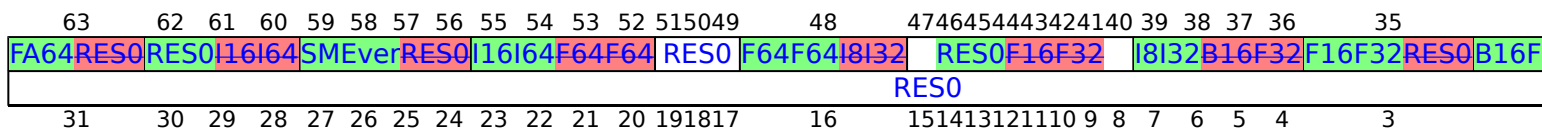
Note

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_AA64SMFR0_EL1 is a 64-bit register.

Field descriptions



FA64, Bits bit [63:56]

Indicates support for execution of the full A64 instruction set when the PE is in Streaming SVE mode. Defined values are:

FA64	Meaning
0b0	Only those A64 instructions defined as being legal can be executed in Streaming SVE mode.
0b1	All implemented A64 instructions can be executed in Streaming SVE mode, when enabled at the current Exception level by SMCR_EL1 .FA64, SMCR_EL2 .FA64, and SMCR_EL3 .FA64.

FEAT_SME_FA64 implements the functionality identified by the value 0b1.

Bits [62:60]

Reserved, RES0.

SMEver, bits [59:56]

Indicates support for SME instructions when [ID_AA64PFR1_EL1](#).SME is not zero. Defined values are:

SMEver	Meaning
0b0000	The non-optional SME instructions are implemented.

All other values are reserved.

FEAT_SME implements the functionality identified by the value 0b0000, when ID_AA64PFR1_EL1.SME is not zero.

From Armv9.2, the only permitted value is 0b0000.

I16I64, bits [55:52]

Indicates SME support for instructions accumulating that 16-bit accumulate integer outer products into 64-bit integer elements in the ZA array tiles. Defined values are:

I16I64	Meaning
0b0000	Instructions that accumulate into 16-bit 64-bit outer integer products elements into in 64-bit the ZA array tiles are not implemented.
0b1111	The variants of the ADDHA, ADDVA, SMOPA, SMOPS, SUMOPA, SUMOPS, UMOPA, UMOPS, USMOPA, and USMOPS instructions that accumulate 16-bit outer products into 64-bit integer element tiles are implemented.

All other values are reserved.

The only permitted values are 0b0000 and 0b1111.

Bits [51:49]

Reserved, RES0.

F64F64, bit [48]

Indicates SME support for instructions accumulating that accumulate into double-precision floating-point elements outer in products the into ZA double-precision array tiles. Defined values are:

F64F64	Meaning
0b0	Instructions that accumulate into double-precision floating-point outer elements products into the double-precision ZA array tiles are not implemented.
0b1	The variants of the FMOPA and FMOPS instructions that accumulate double-precision outer products into double-precision element tiles are implemented.

Bits [47:40]

Reserved, RES0.

I8I32, bits [39:36]

Indicates SME support for accumulating 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 accumulating half-precision floating-point outer products into single-precision floating-point tiles. Defined values are:

F16F32	Meaning
0b0	Instructions that accumulate FP16 outer products into FP32 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 accumulating BFloat16 outer products into 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 BFMOA 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.

Bit [33]

Reserved, RES0.

F32F32, bit [32]

Indicates SME support for accumulating 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.

Accessing ID_AA64SMFR0_EL1

This register is read-only and can be accessed from EL1 and higher.

This register is only accessible from the AArch64 state.

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

MRS <Xt>, ID_AA64SMFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	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() && HCR_EL2.TID3 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            return ID_AA64SMFR0_EL1;
    elseif PSTATE.EL == EL2 then
        return ID_AA64SMFR0_EL1;
    elseif PSTATE.EL == EL3 then
        return ID_AA64SMFR0_EL1;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

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 either or both of [ID_AA64PFR0_EL1](#).SVE and [ID_AA64PFR1_EL1](#).SME are not zero.

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_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								BF16				BitPerm				RES0								AES				SVEver			
31	30	29	28	27	26	25	24	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	FP64 matrix multiplication and related instructions are not implemented.
0b0001	FP64 variant of the FMMLA instruction, and LD1RO* instructions are implemented. The 128-bit element variations of TRN1, TRN2, UZP1, UZP2, ZIP1, and ZIP2 are also implemented.

All other values are reserved.

When the PE is in Streaming SVE mode, software should ignore and treat as zero the value in this field.

FEAT_F64MM implements the functionality identified by 0b0001.

From Armv8.2, the permitted values are 0b0000 and 0b0001.

F32MM, bits [55:52]

Indicates support for the SVE FP32 single-precision floating-point matrix multiplication instruction. Defined values are:

F32MM	Meaning
0b0000	FP32 matrix multiplication instruction is not implemented.
0b0001	FP32 variant of the FMMLA instruction is implemented.

All other values are reserved.

When the PE is in Streaming SVE mode, software should ignore and treat as zero the value in this field.

FEAT_F32MM implements the functionality identified by 0b0001.

From Arm v8.2, the permitted values are 0b0000 and 0b0001.

Bits [51:48]

Reserved, RES0.

I8MM, bits [47:44]

Indicates support for SVE Int8 matrix multiplication instructions. 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.

When the PE is in Streaming SVE mode, software should ignore and treat as zero the value in this field.

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, if FEAT_SVE is implemented, then the only permitted value is 0b0001. Otherwise, the value can be 0b0000 or 0b0001.

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.

When the PE is in Streaming SVE mode, software should ignore and treat as zero the value in this field.

FEAT_SVE_SM4 implements the functionality identified by 0b0001.

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.

When the PE is in Streaming SVE mode, software should ignore and treat as zero the value in this field.

FEAT_SVE_SHA3 implements the functionality identified by 0b0001.

Bits [31:24]

Reserved, RES0.

BF16, bits [23:20]

Indicates support for SVE BFloat16 instructions. Defined values are:

BF16	Meaning
0b0000	BFloat16 instructions are not implemented.
0b0001	BFCVT, BFCVTNT, BFDOT, BFMLALB, BFMLALT, and BFMMMLA instructions are implemented.
0b0010	As 0b0001, but the FPCR .EBF field is also supported.

All other values are reserved.

FEAT_BF16 implements the functionality identified by 0b0001.

FEAT_EBF16 implements the functionality identified by 0b0010.

When Advanced SIMD and SVE are both implemented, this field must return the same value as [ID_AA64ISAR1_EL1](#).BF16.

From Armv8.6, if FEAT_SME is implemented, the permitted values are 0b0001 and 0b0010.

From Armv8.6, if FEAT_SME is not implemented, the only permitted value is 0b0001.

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.

When the PE is in Streaming SVE mode, software should ignore and treat as zero the value in this field.

FEAT_SVE_BitPerm implements the functionality identified by 0b0001.

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 SVE PMULLB and PMULLT instructions with 64-bit source.

All other values are reserved.

When the PE is in Streaming SVE mode, software should ignore and treat as zero the value in this field.

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.

SVEver, bits [3:0]

Indicates support for SVE. Defined values are:

SVEver	Meaning
0b0000	SVE instructions are implemented.
0b0001	The SVE and non-optional SVE2 instructions are implemented.

All other values are reserved.

FEAT_SVE2 and FEAT_SME implement the functionality identified by the value 0b0001.

From Armv9, if FEAT_SME is implemented, the only permitted value is 0b0001. This value indicates that SVE and SVE2 instructions are implemented when the PE is in Streaming SVE mode.

Note

Irrespective of the value of ID_AA64ZFR0_EL1.SVEver, when the PE is in Streaming SVE mode, software should not attempt to execute any of the SVE and SVE2 instructions that are illegal in Streaming SVE mode.

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;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && (!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
        return ID_AA64ZFR0_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64ZFR0_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64ZFR0_EL1;

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(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 also includes support for: <ul style="list-style-type: none"> Extended 16-bit PMEVTYPER<n>.evtCount field. If EL2 is implemented, the HDCR.HPMD control control bit.
0b0101	PMUv3 for Armv8.4. As 0b0100, and adds also includes support for the PMMIR register.
0b0110	PMUv3 for Armv8.5. As 0b0101, and adds also includes support for: <ul style="list-style-type: none"> 64-bit event counters. If EL2 is implemented, the HDCR.HCCD control control bit. If EL3 is implemented, the MDCR_EL3.SCCD control control bit.
0b0111	PMUv3 for Armv8.7. As 0b0110, and adds also includes support for: <ul style="list-style-type: none"> The PMCR.FZO and, if EL2 is implemented, HDCR.HPMFZO controls control bits. If EL3 is implemented, the MDCR_EL3.{MPMX,MCCD} controls control bits.
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.
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.

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

In Armv7, the value 0b0000 can mean that PMUv1 is implemented. PMUv1 is 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:

Support for System registers-based debug model, using registers in the coproc == 0b1110 encoding space, for A and R-profile processors. Defined values are:

CopDBG	Meaning
0b0000	Not supported.
0b0010	Support for Armv6, v6 Debug architecture, with System registers access.
0b0011	Support for Armv6, v6.1 Debug architecture, with System registers access.
0b0100	Support for Armv7, v7 Debug architecture, with System registers access.
0b0101	Support for Armv7, v7.1 Debug architecture, with System registers access.
0b0110	Support for Armv8 debug architecture, with System registers access.
0b0111	Support for Armv8 debug architecture, with System registers access, and Virtualization Host Extensions.
0b1000	Support for Armv8.2 debug architecture, FEAT_Debugv8p2.
0b1001	Support for Armv8.4 debug architecture, FEAT_Debugv8p4.
0b1010	Armv8.8 debug architecture, FEAT_Debugv8p8.

All other values are reserved.

The values 0b0000, 0b0010, 0b0011, 0b0100, and 0b0101 are not permitted in Armv8.

FEAT_VHEFEAT_Debugv8p2 adds the functionality identified by the value 0b01110b1000.

FEAT_Debugv8p2FEAT_Debugv8p4 adds the functionality identified by the value 0b10000b1001.

FEAT_Debugv8p4 addsIn Armv8.0, the functionalityonly identifiedpermitted byvalue the valueis 0b10010b0110.

FEAT_Debugv8p8 addsIn Armv8.1, the functionalityonly identifiedpermitted byvalue the valueis 0b10100b0111.

FromIn Armv8.1Armv8.2, whenthe only permitted value is FEAT_VHE0b1000 is implemented the value 0b0110 is not permitted.

From Armv8.2Armv8.4, the valuesonly permitted value is 0b01100b1001 and 0b0111 are not permitted.

From Armv8.4, the value 0b1000 is not permitted.

From Armv8.8, the value 0b1001 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.

ID_DFR1_EL1, Debug Feature Register 1

The ID_DFR1_EL1 characteristics are:

Purpose

Provides top level information about the debug system in AArch32.

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_DFR1_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ID_DFR1\[31:0\]](#).

Note

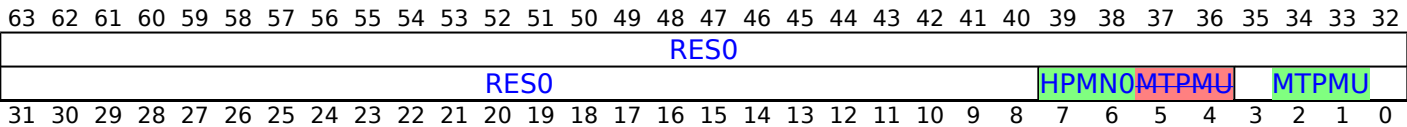
Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

ID_DFR1_EL1 is a 64-bit register.

Field descriptions

When AArch32 is supported:



Bits [63:84]

Reserved, RES0.

HPMN0, bits [7:4]

Zero PMU event counters for a Guest operating system. Defined values are:

HPMN0	Meaning
0b0000	Setting HDCR .HPMN to zero has CONSTRAINED UNPREDICTABLE behavior.
0b0001	Setting HDCR .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.

MTPMU, bits [3:0]

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>. are read/write or RES0.
0b0001	FEAT_MTPMU and FEAT_PMUv3 implemented. PMEVTYPER<n>. are read/write. When FEAT_MTPMU is disabled, the Effective values of PMEVTYPER<n>. are 0.
0b1111	FEAT_MTPMU not implemented. If FEAT_PMUv3 is implemented, PMEVTYPER<n>. 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.

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_DFR1_EL1

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

MRS <Xt>, ID_DFR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0011	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() && (!IsZero(ID_DFR1_EL1) || boolean IMPLEMENTATION_DEFINED "ID_DFR1_EL1ID_DFR1
trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            return ID_DFR1_EL1;
    elsif PSTATE.EL == EL2 then
        return ID_DFR1_EL1;
    elsif PSTATE.EL == EL3 then
        return ID_DFR1_EL1;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The ID_PFR2_EL1 characteristics are:

Purpose

Gives information about the AArch32 programmers' model.

Must be interpreted with [ID PFR0_EL1](#) and [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_PFR2_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ID_PFR2\[31:0\]](#).

Attributes

ID_PFR2_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																
RES0																RAS_frac				SSBS				CSV3								
31	30	29	28	27	26	25	24	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.

RAS_frac, bits [11:8]

RAS Extension fractional field. Defined values are:

RAS_frac	Meaning
0b0000	If ID_PFR0_EL1 .RAS == 0b0001, RAS Extension implemented.
0b0001	If ID_PFR0_EL1 .RAS == 0b0001, as 0b0000 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.

All other values are reserved.

This field is valid only if [ID_PFR0_EL1](#).RAS == 0b0001.

SSBS, bits [7:4]

Speculative Store Bypassing controls in AArch64 state. Defined values are:

SSBS	Meaning
0b0000	AArch32 provides no mechanism to control the use of Speculative Store Bypassing.
0b0001	AArch32 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypass Safe.

In Armv8.0, the permitted values are 0b0000 and 0b0001.

From Armv8.5, the only permitted value is 0b0001.

All other values are reserved.

CSV3, bits [3:0]

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, or 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.

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_PFR2_EL1

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

MRS <Xt>, ID_PFR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0011	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;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TID3 == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            return ID_PFR2_EL1;
    elseif PSTATE.EL == EL2 then
        return ID_PFR2_EL1;
    elseif PSTATE.EL == EL3 then
        return ID_PFR2_EL1;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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.

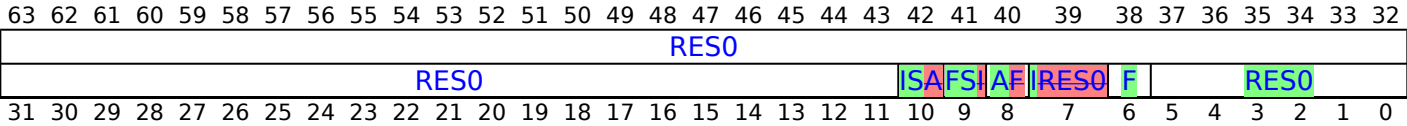
Configuration

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



Bits [63:119]

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.

Note

The function of ISR_EL1.I to indicate the presence of a pending IRQ interrupt is unchanged regardless of Superpriority.

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.

Note

The function of ISR_EL1.F to indicate the presence of a pending FIQ interrupt is unchanged regardless of Superpriority.

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.

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.

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;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.ISR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ISR_EL1;
elseif PSTATE.EL == EL2 then
    return ISR_EL1;
elseif PSTATE.EL == EL3 then
    return ISR_EL1;

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ee9a85f5e501ea

Copyright © 2010-2021 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[51:48]				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 [63:52]

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 ~~defined to be~~ 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] forms the [51:48] upper part of the end physical address of the LORegion value. 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;
elseif 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;
    elseif SCR_EL3.NS == '0' then
        UNDEFINED;
    elseif EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.LOREA_EL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return LOREA_EL1;
elseif PSTATE.EL == EL2 then
    if SCR_EL3.NS == '0' then
        UNDEFINED;
    elseif Halted() && HaveEL(EL3) && EDSCR.SDD == '1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && SCR_EL3.TLOR == '1' then
        UNDEFINED;
    elseif HaveEL(EL3) && SCR_EL3.TLOR == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return LOREA_EL1;
elseif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        UNDEFINED;
    else
        return 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() && (!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];
    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];
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        else
            LOREA_EL1 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd5b36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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																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	
																Valid																

Any of the fields in this register are permitted to be cached in a TLB.

Bits [63:52]

Reserved, RES0.

SA, bits [51:16]

SA encoding when FEAT_LPA is implemented

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 [35:0]

~~When 52-bit addresses are not in use, SA[35:32] is RES0.~~

Bits [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 defined to be 0x0000.

~~For When implementations 52-bit with addresses fewer are than in 52 use, physical SA[35:32] address forms bits, the corresponding upper bits part of this the field address are value, 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

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				SA																															

Bits [35:32]

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 defined to be 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() && (!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
            return 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
                return LORSA_EL1;
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        else
            return 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() && (!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];
    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];
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            UNDEFINED;
        else
            LORSA_EL1 = X[t];

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(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\[31:0\]](#).

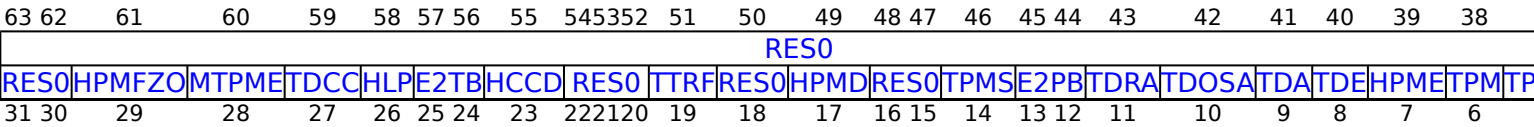
If EL2 is not implemented, this register is RES0 from EL3.

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



Bits [63:37]

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_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 event counters in the range [0 .. (MDCR_EL2.HPMN-1)] and [PMCCNTR_EL0](#).~~

~~This field [MDCR_EL2.HPMN](#) does is not equal affect the operation of other event counters and to [PMCCNTR_EL0](#)[PMCR_EL0](#).N, this field has no effect.~~

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_ELO [(PMCR_ELO .N-1):MDCR_EL2.HPMN] is nonzero.

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 event counters in the range [0 .. (MDCR_EL2.HPMN-1)] and [PMCCNTR_ELO](#).

The operation of this field ignores the values of [PMOVSLR_ELO](#)[(MDCR_EL2.HPMN-1):0].

This field [MDCR_EL2.HPMN](#) does not affect the operation of other event counters and to [PMCCNTR_ELO](#) [PMCR_ELO](#).N, this field has no effect.

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>_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 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..(or [PMCR.N](#), this bit affects the operation of event counters in the range [MDCR_EL2.HPMN..[PMCR_EL0.N-1](#)]] or [MDCR_EL2.HPMN..[PMCR.N-1](#)]]. Otherwise this bit has no effect on the operation of the event counters.

Note

The effect of MDCR_EL2.HPMN on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state.

For more information see the description of the MDCR_EL2.HPMN field.

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 TraceBufferEnabled() == TRUE, 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 TraceBufferEnabled() == TRUE, 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 TraceBufferEnabled() == TRUE, 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 TRFCR_EL1 and TRFCR at EL1 are not affected by this control.
0b1	Accesses to TRFCR_EL1 and TRFCR at EL1 generate a trap exception to EL2 when EL2 is enabled in the current Security state.

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 this MDCR_EL2.HPMN is not 0, this field affects applies the only operation of event counters in the range [0 .. (MDCR_EL2.HPMN-1)]. to:

- The event counters in the range [0 .. (MDCR_EL2.HPMN-1)].
- If [PMCR_EL0](#).DP is 1, [PMCCNTR_EL0](#).

The other event counters are not affected. When [PMCR_EL0.DP](#) is 0, [PMCCNTR_EL0](#) is not affected.

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_EL0 are not affected by this mechanism.
0b1	If ExternalSecureNoninvasiveDebugEnabled() is FALSE, event counting by some event counters is prohibited at EL2, and if PMCR_EL0.DP is 1, PMCCNTR_EL0 is disabled at EL2.

If ExternalSecureNoninvasiveDebugEnabled() is TRUE, this field does not affect the event counters and does not affect [PMCCNTR_EL0](#). ~~are not affected by this field.~~

Otherwise, this field applies only to:

- If [MDCR_EL2.HPMN](#) is not 0, this field affects the operation of The 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, [PMCCNTR_EL0](#).~~
- If [PMCR_EL0.DP](#) is 1, this field affects [PMCCNTR_EL0](#).

The other event counters are not affected. When [PMCR_EL0.DP](#) is 0, [PMCCNTR_EL0](#) is not affected.

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 [PMSLATR_EL1](#).
- If FEAT_SPEv1p2 is implemented, [PMSNEVER_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 DBGDSCRext .UDCCdis or 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](#), [DBGOSECRR](#).
 - 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](#)] or [HDCR_EL2.HPMN..[PMCR_EL0.N-1](#)], are enabled and disabled by this bit. Otherwise this bit has no effect on the operation of the event counters.

Note

The effect of MDCR_EL2.HPMN on the operation of this bit applies regardless of whether EL2 is enabled in the current Security state.

For more information see the description of the HPMN field.

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](#), [PMOVSCCLR_EL0](#), [PMSWINC_EL0](#), [PMSELR_EL0](#), [PMCEID0_EL0](#), [PMCEID1_EL0](#), [PMCCNTR_EL0](#), [PMXEVTYPER_EL0](#), [PMXEVCNTR_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](#), [PMOVS](#), [PMSWINC](#), [PMSELR](#), [PMCEID0](#), [PMCEID1](#), [PMCCNTR](#), [PMXEVTYPER](#), [PMXEVCNTR](#), [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 PMCR_EL0 or PMCR are trapped to EL2 when EL2 is enabled in the current Security state, unless it is trapped by PMUSERENR .EN or 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 that are accessible from EL3, EL2, EL1, and from EL0 if permitted.

If HPMN is not 0 and is less than `PMCR_ELO.N`, HPMN divides the Performance Monitors into two first ranges: range `[0..(HPMN-1)]`, and a second range `[HPMN..(PMCR_ELO.N-1)]`.

If for an event counter in the range `[0..(HPMN-1)]`: FEAT_HPMN0 is implemented and this field is 0, all events counters are in the second range and none are in the first range.

If HPMN is equal to `PMCR_ELO.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 EL3, EL2, and EL1, and from EL0 if permitted by `PMUSERENR_ELO` or `PMUSERENR`.
- The counter is accessible from EL0 if permitted by `PMUSERENR_ELO` or `PMUSERENR`.
- If FEAT_PMUv3p5 is implemented, `PMCR_ELO.LP` or `PMCR.LP` determines whether the counter overflow flag is set on unsigned overflow of `PMEVCNTR<n>_ELO[31:0]` or `PMEVCNTR<n>_ELO[63:0]`.
- The counter is enabled by `PMCR_ELO.E` or `PMCR.E` and bit <n> of `PMCNTENSET_ELO[n]` enable the operation of event counter n.

Note

If HPMN is equal to `PMCR_ELO.N`, this applies to all event counters.

If HPMN is less than `PMCR_ELO.N`, for an event counter in the range `[HPMN..(PMCR_ELO.N-1)]`:

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_ELO`.
- If FEAT_SEL2 is disabled or is not implemented, the counter is also accessible from Secure EL1, and from Secure EL0 if permitted by `PMUSERENR_ELO`.
- If FEAT_PMUv3p5 is implemented, `MDCR_EL2.HLP` determines whether the counter overflow flag is set on unsigned overflow of `PMEVCNTR<n>_ELO[31:0]` or `PMEVCNTR<n>_ELO[63:0]`.
- The counter is enabled by `MDCR_EL2.HPME` and bit <n> of `PMCNTENSET_ELO[n]` enable the operation of event counter n.

If HPMN this field is set to 0, or to a value larger than `PMCR_ELO.N`, or then if the following 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.
- 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_ELO.N`.
 - All counters are reserved for EL2 and EL3 use, meaning no counters are accessible from EL1 and EL0.

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
        return MDCR_EL2;
elsif PSTATE.EL == EL3 then
    return 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];
elsif PSTATE.EL == EL3 then
    MDCR_EL2 = X[t];

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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

AArch64 System register MDCR_EL3 bits [31:0] can be mapped to AArch32 System register [SDCR\[31:0\]](#), but this is not architecturally mandated.

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

63	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																							E3B		
RES0	MTPME	TDCC	NSTBE	NSTB	SCCD	ETAD	EPMA	EDAD	TTRF	STE	SPME	SDD	SPD32	NSPB	NSPBE	TDOSA	TDARESO								
31	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 [63:39]37]

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. In conjunction with MDCR_EL3.SPME, controls when event counters are enabled at EL3 and in other Secure Exception levels.

MPMX	Meaning
0b0	Event counting and PMCCNTR_EL0 are not affected by this mechanism.
0b1	Event counting by some or all event counters is prohibited at EL3. If PMCR_EL0.DP is 1, PMCCNTR_EL0 is disabled at EL3. Otherwise, PMCCNTR_EL0 is not affected by this mechanism.

If EL2 is implemented, MDCR_EL3.SPME == 1, and MDCR_EL2.HPMN is less than PMCR_EL0.N then all the following are true:

- If this field affects the operation of event counters in the range [0 .. (MDCR_EL2.HPMN-1)] is at not 0EL3, this field affects the operation of event counters in the range [0 .. (if.DP is 1, the operation of PMCCNTR_EL0MDCR_EL2PMCR_EL0.HPMN-1)] at EL3.
- This field does not affect the operation of other event counters in the range [MDCR_EL2.HPMN .. (PMCR_EL0.N-1)].
- If this applies even when EL2 is disabled in Secure state. PMCR_EL0.DP is 1, this field affects the operation of PMCCNTR_EL0 at EL3.

The operation of this field applies even when EL2 is disabled in the current Security state.

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.

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.

Bits [31:29]

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: [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_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.

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 PE Trace Unit registers is implemented and FEAT_TRBE is implemented:

External Trace Access Disable. Together with MDCR_EL3.ETADE, controls access to PE Trace Unit registers by an external debugger.

ETADE	ETAD	Meaning
0b0	0b0	Access to PE Trace Unit registers by an external debugger is permitted.
0b0	0b1	Root and Secure access to PE Trace Unit registers by an external debugger is permitted. Realm and Non-secure access to PE Trace Unit registers by an external debugger is not permitted.
0b1	0b0	Root and Realm access to PE Trace Unit registers by an external debugger is permitted. Secure and Non-secure access to PE Trace Unit registers by an external debugger is not permitted.
0b1	0b1	Root access to PE Trace Unit registers by an external debugger is permitted. Secure, Non-secure, and Realm access to PE 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 PE Trace Unit registers is implemented and FEAT_TRBE is implemented:

External Trace Access Disable. Controls Non-secure access to PE Trace Unit registers by an external debugger.

ETAD	Meaning
0b0	Non-secure accesses from an external debugger to PE Trace Unit are allowed.
0b1	Non-secure accesses from an external debugger to some PE 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.

EPMADE, 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.

Otherwise:

Reserved, RES0.

STE, bit [18]**When FEAT_TRF 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 event counting in Secure state and EL3.

SPME	Meaning
0b0	When MDCR_EL3.MPMX == 0: Event counting is prohibited in Secure state. If PMCR_EL0.DP is 1, PMCCNTR_EL0 is disabled in Secure state. Otherwise, PMCCNTR_EL0 is not affected by this mechanism.
0b1	When MDCR_EL3.MPMX == 0: Event counting and PMCCNTR_EL0 are not affected by this mechanism.

When MDCR_EL3.MPMX is 0, 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 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.

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 and FEAT_Debugv8p2 is implemented:

Secure Performance Monitors Enable. Controls event counting in Secure state.

SPME	Meaning
0b0	Event counting is prohibited in Secure state. If PMCR_EL0.DP is 1, PMCCNTR_EL0 is disabled in Secure state. Otherwise, PMCCNTR_EL0 is not affected by this mechanism.
0b1	Event counting 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 event counting in Secure state.

SPME	Meaning
0b0	If <code>ExternalSecureNoninvasiveDebugEnabled()</code> 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]

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.

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.

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](#), [PMSFCR_EL1](#), [PMSICR_EL1](#), [PMSIDR_EL1](#), [PMSIRR_EL1](#), and [PMSLATFR_EL1](#).
- If FEAT_SPEv1p2 is implemented, [PMSNEVER_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, [PMSNEVER_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_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.

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](#).
- 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](#), [DBGWFAR](#), [DBGVCR](#), [DBGBVR<n>](#), [DBGBCR<n>](#), [DBGBXVR<n>](#), [DBGWCR<n>](#), [DBGWVR<n>](#).
 - [DBGCLAIMSET](#), [DBGCLAIMCLR](#), [DBGAUTHSTATUS](#), [DBGDEVID](#), [DBGDEVID1](#), [DBGDEVID2](#), [DBGOSECCR](#).
- 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	This control does not cause any instructions to be trapped.
0b1	EL0, EL1, and EL2 accesses to the debug registers, other than the registers that can be trapped by MDCR_EL3.TDOSA, are trapped to EL3, from any Security state and both Execution states, unless it is trapped by DBGDSCRExt .UDCCdis, MDSCR_EL1 .TDCC, HDCR .TDA or MDCR_EL2 .TDA.

The reset behavior of this field is:

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

Bits [8:7]

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](#), [PMOVSLR_EL0](#), [PMSWINC_EL0](#), [PMSELR_EL0](#), [PMCEID0_EL0](#), [PMCEID1_EL0](#), [PMCCNTR_EL0](#), [PMXEVTYPER_EL0](#), [PMXEVCNTR_EL0](#), [PMUSERENR_EL0](#), [PMINTENSET_EL1](#), [PMINTENCLR_EL1](#), [PMOVSSSET_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](#), [PMOVSR](#), [PMSWINC](#), [PMSELR](#), [PMCEID0](#), [PMCEID1](#), [PMCCNTR](#), [PMXEVTYPER](#), [PMXEVCNTR](#), [PMUSERENR](#), [PMINTENSET](#), [PMINTENCLR](#), [PMOVSSSET](#), [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 PE Trace Unit registers is implemented and FEAT_TRBE is implemented:

External Trace Access Disable Extended. Together with MDCR_EL3.ETAD, controls access to PE 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.

Root Trace enable. Enables tracing in Root state.

RTTE	Meaning
0b0	Trace prohibited in Root state, unless overridden by the IMPLEMENTATION DEFINED authentication interface.
0b1	Trace in Root 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:**Bit [1]****RTTE, bit [1]**

When FEAT_RME is implemented and FEAT_TRF is implemented:

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
    return 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];
```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(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	

The upper part of the address value.

If the physical address size in bits (PAsize) is less than 52, then the register bits corresponding to ROMADDR[39:PAsize] are RES0.

The reset behavior of this field is:

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

ROMADDR encoding when FEAT_LPA is not implemented and/or MDRAR_EL1.ValidAArch32 != 0b00 supported

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	ROMADDR

Bits [39:36]

Reserved, RES0.

ROMADDR, bits [35:0]

Bits [39:12] of the ROM table physical address.

Bits [11:0] of the ROM table physical address are defined to be 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 memory. It is IMPLEMENTATION DEFINED whether the ROM table is also accessible in Secure memory.

Arm strongly recommends that bits ROMADDR[(PAsize-1):32] are zero in any system that supports AArch32 at the highest implemented Exception level.

If MDRAR_EL1.Valid == 0b00, then this field is UNKNOWN.

The reset behavior of this field is:

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

ROMADDR encoding when MDRAR_EL1.Valid == 0b00

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	
	UNKNOWN

Bits [39:0]

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
            return 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
            return MDRAR_EL1;
elsif PSTATE.EL == EL3 then
    return MDRAR_EL1;

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ca

Copyright © 2010-2021 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\[5:2\]](#).

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																															
TFO	RXfull	TXfull	RES0	RXOT	TXU	RES0	INTdis	TDARE	RES0	SC2	RAZ/WI	MDE	HDE	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 [63:32]

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() && (!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,NV1,NV> == '1x1' then
        return NVMem[0x158];
    else
        return 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
        return MDSCR_EL1;
elsif PSTATE.EL == EL3 then
    return 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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDBGWTR_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,NV1,NV> == '1x1' then
        NVMem[0x158] = X[t];
    else
        MDSCR_EL1 = X[t];
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];
elsif PSTATE.EL == EL3 then
    MDSCR_EL1 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdff36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MPAMSM_EL1, MPAM Streaming Mode Register

The MPAMSM_EL1 characteristics are:

Purpose

Holds information to generate MPAM labels for memory requests issued by SME, SVE, and SIMD&FP load and store instructions and, when the PE is in Streaming SVE mode, by SVE and SIMD&FP load and store instructions. For those requests, the MPAM labels in this register have precedence over the labels in MPAM0_EL1, MPAM1_EL1, MPAM2_EL2, and MPAM3_EL3.

It is IMPLEMENTATION DEFINED whether the MPAM labels in this register are used for memory requests due to hardware page table walks or and page table updates performed as a result of SME, load SVE, and SIMD&FP load/store instructions, and SVE prefetch instructions, when the PE is in Streaming SVE mode, SVE and SIMD&FP load and store instructions, and SVE prefetch instructions. mode.

The MPAM labels in this register are only used if MPAM1_EL1.MPAMEN is 1.

For memory requests issued from EL0, the MPAM PARTID in this register is virtual and mapped into a physical PARTID when all of the following are true:

- EL2 is implemented and enabled in the current Security state, and HCR_EL2.{E2H, TGE} != {1, 1}.
- The MPAM virtualization option is implemented and MPAMHCR_EL2.EL0_VPMEN == 1.

For memory requests issued from EL1, the MPAM PARTID in this register is virtual and mapped into a physical PARTID when all of the following are true:

- EL2 is implemented and enabled in the current Security state.
- The MPAM virtualization option is implemented and MPAMHCR_EL2.EL1_VPMEN == 1.

Configuration

This register is present only when FEAT_MPAM is implemented and FEAT_SME is implemented. Otherwise, direct accesses to MPAMSM_EL1 are UNDEFINED.

Attributes

MPAMSM_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																PMG_D								RES0							
PARTID_D																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 [63:48]

Reserved, RES0.

PMG_D, bits [47:40]

Performance monitoring group property for PARTID_D.

The reset behavior of this field is:

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

Bits [39:32]

Reserved, RES0.

PARTID_D, bits [31:16]

Partition ID for requests data issued accesses due to the execution of any SME, Exception SVE, level and of SIMD&FP SME load and store instructions and, performed when the PE is in Streaming SVE mode, SVE at and any SIMD&FP Exception load and store instructions and SVE prefetch instructions. level.

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 MPAMSM_EL1

None of the fields in this register are permitted to be cached in a TLB.

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

MRS <Xt>, MPAMSM_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HaveEL(EL3) && MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && MPAM2_EL2.EnMPAMSM == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return MPAMSM_EL1;
elsif PSTATE.EL == EL2 then
    if HaveEL(EL3) && MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return MPAMSM_EL1;
elsif PSTATE.EL == EL3 then
    return MPAMSM_EL1;

```

MSR MPAMSM_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1010	0b0101	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if HaveEL(EL3) && MPAM3_EL3.TRAPLOWER == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        elsif EL2Enabled() && MPAM2_EL2.EnMPAMSM == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            MPAMSM_EL1 = X[t];
    elsif PSTATE.EL == EL2 then
        if HaveEL(EL3) && MPAM3_EL3.TRAPLOWER == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                MPAMSM_EL1 = X[t];
    elsif PSTATE.EL == EL3 then
        MPAMSM_EL1 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd5b36e47856e443a7ee9a85f5e501ea

Copyright © 2010-2021 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\[63:0\]](#).

Attributes

PAR_EL1 is a 64-bit register.

Field descriptions

When PAR_EL1.F == 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
ATTR								RES0				PA[51:48]				PA[47:12]																	
PA[47:12]												NSE		IMPLEMENTATION DEFINED						NS		SH		RES0						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 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](#).

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	Secure.
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 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.

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 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.

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 PAR_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	38	37	36	35	34	33	32								
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.

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 [47:12]

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.	
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
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
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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGTR_EL2.PAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return PAR_EL1;
elsif PSTATE.EL == EL2 then
    return PAR_EL1;
elsif PSTATE.EL == EL3 then
    return PAR_EL1;

```

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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HFGWTR_EL2.PAR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        PAR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    PAR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    PAR_EL1 = X[t];

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

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.

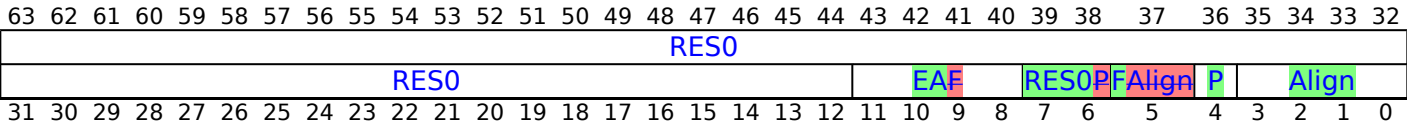
Configuration

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



Bits [63:126]

Reserved, RES0.

EA, bits [11:8]

External Abort handling. Describes how the PE manages External aborts on writes made by the Statistical Profiling Extension to the Profiling Buffer.

EA	Meaning
0b0000	Not described.
0b0001	The PE ignores External aborts on writes made by the Statistical Profiling Extension.
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.

Access to this field is RO.

Bits [7:6]

Reserved, RES0.

F, bit [5]

Flag updates. Describes how whether the address translation performed by the Statistical Profiling Extension Buffer manages the Access flag and dirty state. Defined values are:

F	Meaning
0b0	Hardware management of the Access flag Flag and dirty state for accesses made by the Statistical Profiling Extension is always disabled for all translation stages.
0b1	Hardware management off for the Access flag Flag and dirty state for accesses made by the Statistical Profiling Extension is controlled in the same way as explicit memory accesses in the Profiling Buffer owning translation regime.

If hardware management of the Access Flag is disabled for a stage of translation, an access to 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 might generate a Permission fault, depending on the values of the access permission bits in the descriptor.

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.

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**. **If this field is non-zero, then the PE must pad every record up to a multiple of this size.** Defined values are:

Align	Meaning
0b0000	Byte.
0b0001	Halfword.
0b0010	Word.
0b0011	Doubleword.
0b0100	16 bytes.Bytes.
0b0101	32 bytes.Bytes.
0b0110	64 bytes.Bytes.
0b0111	128 bytes.Bytes.
0b1000	256 bytes.Bytes.
0b1001	512 bytes.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.

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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMBIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return PMBIDR_EL1;
elsif PSTATE.EL == EL2 then
    return PMBIDR_EL1;
elsif PSTATE.EL == EL3 then
    return PMBIDR_EL1;
```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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															
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 [63:32]

Reserved, RES0.

EC, bits [31:26]

Exception 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 Exception 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 MSS encoding for other 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 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 Extension.

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.	
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.	
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

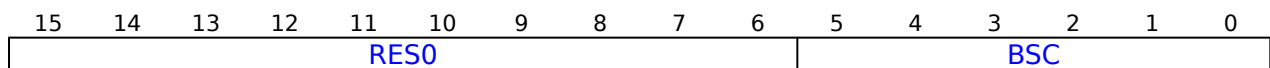
All other values are reserved.

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



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:

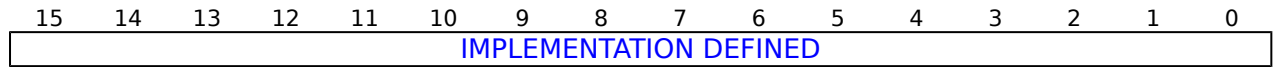
- On a Warm reset, this field resets to an architecturally UNKNOWN value.

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() && (!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,NV1,NV> == '1x1' then
        return NVMem[0x820];
    else
        return 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
        return PMBSR_EL1;
elseif PSTATE.EL == EL3 then
    return 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() && (!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,NV1,NV> == '1x1' then
            NVMem[0x820] = X[t];
        else
            PMBSR_EL1 = X[t];
    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];
    elsif PSTATE.EL == EL3 then
        PMBSR_EL1 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd536e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCEID0_EL0, Performance Monitors Common Event Identification register 0

The PMCEID0_EL0 characteristics are:

Purpose

Defines which **Commoncommon** architectural events and **Commoncommon** microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0000 to 0x001F and 0x4000 to 0x401F.

For more information about the **Commoncommon** 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 [PMCEID0\[31:0\]](#).

AArch64 System register PMCEID0_EL0 bits [63:32] are architecturally mapped to External register [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
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 **Commoncommon** event (0x4000 + n).

For each bit:

IDhi<n>	Meaning
0b0	The Commoncommon event is not implemented, or not counted.
0b1	The Commoncommon event is implemented.

When the value of a bit in the field is 1, the corresponding **Commoncommon** event is implemented and counted.

Note

Arm recommends that if a **Commoncommon** 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 **Commoncommon** 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 **Commoncommon** event n.

For each bit:

ID<n>	Meaning
0b0	The Commoncommon event is not implemented, or not counted.
0b1	The Commoncommon event is implemented.

When the value of a bit in the field is 1, the corresponding **Commoncommon** event is implemented and counted.

Note

Arm recommends that if a **Commoncommon** 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 **Commoncommon** 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' && (!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
                return 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() && (!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
                    return 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
                        return PMCEID0_EL0;
            elsif PSTATE.EL == EL3 then
                return PMCEID0_EL0;

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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 **Commoncommon** architectural events and **Commoncommon** microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0020 to 0x003F and 0x4020 to 0x403F.

For more information about the **Commoncommon** 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 [PMCEID1\[31:0\]](#).

AArch64 System register PMCEID1_EL0 bits [63:32] are architecturally mapped to External register [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 **Commoncommon** event (0x4020 + n).

For each bit:

IDhi<n>	Meaning
0b0	The Commoncommon event is not implemented, or not counted.
0b1	The Commoncommon event is implemented.

When the value of a bit in the field is 1, the corresponding **Commoncommon** event is implemented and counted.

Note

Arm recommends that if a **Commoncommon** 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 **Commoncommon** 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 **Commoncommon** event (0x0020 + n).

For each bit:

ID<n>	Meaning
0b0	The Commoncommon event is not implemented, or not counted.
0b1	The Commoncommon event is implemented.

When the value of a bit in the field is 1, the corresponding **Commoncommon** event is implemented and counted.

Note

Arm recommends that if a **Commoncommon** 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 **Commoncommon** 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' && (!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
            return 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() && (!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
            return 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
            return PMCEID1_EL0;
    elsif PSTATE.EL == EL3 then
        return PMCEID1_EL0;

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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 [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.

FZS	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 EL2 is implemented, then:

In the description of this field:

- This field affects the operation of event counters in the range [0 .. ([MDCR_EL2](#).HPMN-1)].

If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.

- If [MDCR_EL2](#).HPMN is less than PMCR_EL0.N:
 - This field does not affect the operation of event counters in the range [[MDCR_EL2](#).HPMN .. (PMCR_EL0.N-1)].

If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.

- This applies even when EL2 is disabled in the current Security state.

If EL2 is not implemented, PMN is PMCR_EL0.N.

This field does not affect the operation of PMCCNTR_EL0.

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 != 0b000000000x00:

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.

FZO	Meaning
0b0	Do not freeze on overflow.
0b1	Event counters do not count when PMOVSCLR_EL0 [(N-1):0] is nonzero, where N is the value of MDCR_EL2 .HPMN if EL2 is implemented, and PMCR_EL0 .N otherwise.

If EL2 is implemented, then:

In the description of this field:

- This field affects the operation of event counters in the range [0 .. ([MDCR_EL2](#).HPMN-1)].
If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.
- If [MDCR_EL2](#).HPMN is less than [PMCR_EL0](#).N:
 - This field does not affect the operation of event counters in the range [[MDCR_EL2](#).HPMN .. ([PMCR_EL0](#).N-1)].
 - The operation of this field ignores the values of [PMOVSCLR_EL0](#)[([PMCR_EL0](#).N-1):[MDCR_EL2](#).HPMN].
 If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.
- This applies even when EL2 is disabled in the current Security state.
If EL2 is not implemented, PMN is [PMCR_EL0](#).N.

This field does not affect the operation of [PMCCNTR_EL0](#).

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](#)_{EL2} is not implemented 0, this field affects the operation of event counters in the range [0 .. (PMN-1)], and [MDCR_EL2.HPMN](#) or [HDCR.HPMN](#) is less than [PMCR_EL0.N](#), this bit does not affect the operation of event counters in the range [[HDCR.HPMN](#)..([PMCR_EL0.N](#)-1)] or [[MDCR_EL2.HPMN](#)..([PMCR_EL0.N](#)-1)].

Note

The effect of [MDCR_EL2.HPMN](#) or [HDCR.HPMN](#) on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state. For more information, see the description of [MDCR_EL2.HPMN](#) or [HDCR.HPMN](#).

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	When event counting is prohibited, cycle counting by PMCCNTR_EL0 is disabled in prohibited regions: <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 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. 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.

For more information see 'Prohibiting event 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 PE 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. **The effects of writing to this bit are:**

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 Reset, all event counters accessible in the range of Exception affected event counters level, resets not each event counter including PMEVCNTR<n>_EL0 and PMCCNTR_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)]. [MDCR_EL2.HPMN](#) is less than [MDCR_EL2.HPMN..\(PMCR_EL0.N-1\)](#).
- If EL2 is not implemented, EL2 is disabled in the current Security state, a write of 1 to this bit resets all the event counters. or [MDCR_EL2.HPMN](#) equals [PMCR_EL0.N](#), 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](#).

In EL2 and EL3, a write of 1 to this bit resets all the event counters.

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.

E	Meaning
0b0	All event counters in the range [0..(PMN-1)] and PMCCNTR_EL0 , are disabled.
0b1	All event counters in the range [0..(PMN-1)] and PMCCNTR_EL0 , are enabled by PMCNTENSET_EL0 .

If EL2 is implemented, then:

- If EL2 is using AArch32, PMN is [HDCR.HPMN](#).
- If EL2 is using AArch64, PMN is [MDCR_EL2.HPMN](#).
- If PMN is less than [PMCR_EL0.N](#), this bit does not affect the operation of event counters in the range [PMN..(PMCR_EL0.N-1)].

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](#).

Note

The effect of [MDCR_EL2.HPMN](#) or [HDCR.HPMN](#) on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state. For more information, see the description of [MDCR_EL2.HPMN](#) or [HDCR.HPMN](#).

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
        return 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
        return 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
        return PMCR_EL0;
elsif PSTATE.EL == EL3 then
    return 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' && (!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];
    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() && (!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];
    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];
    elsif PSTATE.EL == EL3 then
        PMCR_EL0 = X[t];

```

3020/09/2021 1412:5336; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)htmldiff from-(new)

PMEVCNTR<n>_EL0, Performance Monitors Event Count Registers, n = 0 - 30

The PMEVCNTR<n>_EL0 characteristics are:

Purpose

Holds event counter n, which counts events, where n is 0 to 30.

Configuration

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

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

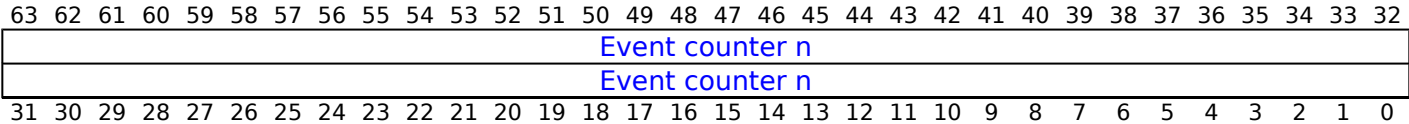
This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMEVCNTR<n>_EL0 are UNDEFINED.

Attributes

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

Field descriptions

When FEAT_PMUv3p5 is implemented:



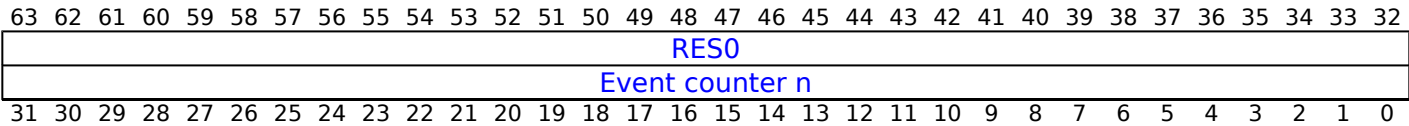
Bits [63:0]

Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

The reset behavior of this field is:

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

Otherwise:



Bits [63:32]

Reserved, RES0.

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<n>_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b10:n[4:3]	n[2:0]

```

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' && (!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 HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMEVCNTR_EL0[UInt(CRm<1:0>:op2<2:0>)];
    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() && (!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 HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMEVCNTR_EL0[UInt(CRm<1:0>:op2<2:0>)];
    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
            return PMEVCNTR_EL0[UInt(CRm<1:0>:op2<2:0>)];
    elsif PSTATE.EL == EL3 then
        return PMEVCNTR_EL0[UInt(CRm<1:0>:op2<2:0>)];

```

MSR PMEVCNTR<n>_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b10:n[4:3]	n[2:0]

```

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' && (!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 HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMEVCNTR_EL0[UInt(CRm<1:0>:op2<2:0>)] = X[t];
    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() && (!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 HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMEVCNTR_EL0[UInt(CRm<1:0>:op2<2:0>)] = X[t];
    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(CRm<1:0>:op2<2:0>)] = X[t];
    elsif PSTATE.EL == EL3 then
        PMEVCNTR_EL0[UInt(CRm<1:0>:op2<2:0>)] = X[t];

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

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

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

AArch64 System register PMEVTYPER<n>_EL0 bits [63:32] are architecturally mapped to External register [PMEVFILTR<n>\[31:0\]](#).

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	
TCRES0			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 bits [63:6132]

When 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.

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.
0b001	Not-equal, count. The counter increments by 1 on each processor cycle when V is not equal to PMEVTYPER<n>_EL0.TH.
0b010	Equals. The counter increments by V on each processor cycle when V is equal to PMEVTYPER<n>_EL0.TH.
0b011	Equals, count. The counter increments by 1 on each processor cycle when V is equal to PMEVTYPER<n>_EL0.TH.
0b100	Greater-than-or-equal. The counter increments by V on each processor cycle when V is PMEVTYPER<n>_EL0.TH or more.
0b101	Greater-than-or-equal, count. The counter increments by 1 on each processor cycle when V is PMEVTYPER<n>_EL0.TH or more.
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.

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.

Bits [60:44]

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 $2^{(PMMIR_EL1.THWIDTH)}$.

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 in Transactional state. The possible values of this bit are:

T	Meaning
0b0	This bit has no effect on the filtering of events.
0b1	Do not count events in Transactional state.

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 Software event must number program of this the field with an event that is counted supported by event the counter PE being programmed. 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 `evtCount` is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written: `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.

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 `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.
- For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the `evtCount` field is UNKNOWN.

Note

UNPREDICTABLE means the event must not expose privileged information.

Arm recommends that for all values that represent reserved or unsupported events, 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.

Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on `evtCount` is the value written.

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>, PMEVTYPEPER<n>_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b11:n[4:3]	n[2:0]

```

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' && (!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 HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMEVTYPEPER_EL0[UInt(CRm<1:0>:op2<2:0>)];
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() && (!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 HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMEVTYPEPER_EL0[UInt(CRm<1:0>:op2<2:0>)];
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
        return PMEVTYPEPER_EL0[UInt(CRm<1:0>:op2<2:0>)];
elsif PSTATE.EL == EL3 then
    return PMEVTYPEPER_EL0[UInt(CRm<1:0>:op2<2:0>)];

```

MSR PMEVTYPEPER<n>_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b11:n[4:3]	n[2:0]


```

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' && (!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 HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMEVTYPER_EL0[UInt(CRm<1:0>:op2<2:0>)] = X[t];
    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() && (!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 HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMEVTYPER_EL0[UInt(CRm<1:0>:op2<2:0>)] = X[t];
    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[UInt(CRm<1:0>:op2<2:0>)] = X[t];
    elsif PSTATE.EL == EL3 then
        PMEVTYPER_EL0[UInt(CRm<1:0>:op2<2:0>)] = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(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

63	62	61	60	59	58	57	56	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								THWIDTH				BUS_WIDTH				BUS_WIDTH				BUS_SLOTS				BUS_SLOTSSLOTS				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				

Bits [63:24:20]

Reserved, RES0.

THWIDTH, bits [23:20]

PMEVTYPER<n>_EL0.TH width. Indicates implementation of the FEAT_PMUv3_TH feature, and, if implemented, the size of the **PMEVTYPER<n>_EL0.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.

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. Defined values are:

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.

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.

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**. **RAZ**.

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;
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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMMIR_EL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && MDCR_EL2.TPM == '1' then
        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
        return PMMIR_EL1;
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
        return PMMIR_EL1;
elseif PSTATE.EL == EL3 then
    return PMMIR_EL1;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMSIDR_EL1, Sampling Profiling ID Register

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				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. ~~Defined values are:~~

CountSize	Meaning	Applies when
0b0010	12-bit saturating counters.	
0b0011	16-bit saturating counters.	From Armv8.8

All other values are reserved.

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. 1024 bytes
0b1011	2KB.

All other values are reserved.

The values 0b0100 and 0b0101 are not permitted for an implementation.

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.

Access to this field is **RO**.

Bit [7]

Reserved, RES0.

FnE, bit [6]

Filtering by events, inverted. Defined values are:

FnE	Meaning
0b0	PMSNEVFR_EL1 is not implemented and PMSFCR_EL1 .FnE is RES0.
0b1	PMSNEVFR_EL1 and PMSFCR_EL1 .FnE are implemented.

FEAT_SPEv1p2The addvalue the1 functionalityindicates identifiedsupport byfor the valueFEAT_SPEv1p21.feature.

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.

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.

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.

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;
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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMSIDR_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
            return PMSIDR_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
            return PMSIDR_EL1;
elsif PSTATE.EL == EL3 then
    return PMSIDR_EL1;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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

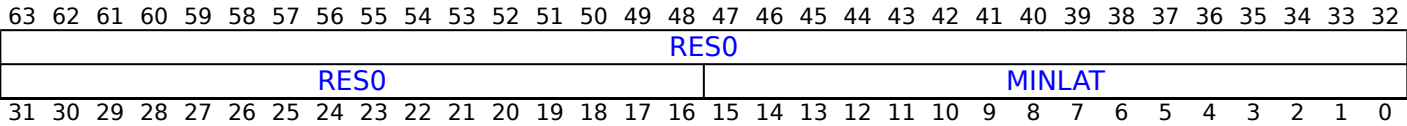
Configuration

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:**16:12**]

Reserved, RES0.

MINLAT, bits [**15:11**:0]

Minimum latency. When ~~PMSFCR_EL1.FL == 1, defines the minimum total latency for filtered operations. Samples with a total latency less than MINLAT will not be recorded~~ **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 this field is ignored by the PE when PMSFCR_EL1.FL == 0.~~ **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() && (!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,NV1,NV> == '1x1' then
            return NVMem[0x848];
        else
            return 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
            return PMSLATFR_EL1;
    elsif PSTATE.EL == EL3 then
        return 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() && (!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,NV1,NV> == '1x1' then
            NVMem[0x848] = X[t];
        else
            PMSLATFR_EL1 = X[t];
    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];
    elsif PSTATE.EL == EL3 then
        PMSLATFR_EL1 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbfdb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMXEVTYPER_EL0, Performance Monitors Selected Event Type Register

The PMXEVTYPER_EL0 characteristics are:

Purpose

When [PMSELR_EL0.SEL](#) selects an event counter, this accesses a [PMEVTYPER<n>_EL0](#) register. When [PMSELR_EL0.SEL](#) selects the cycle counter, this accesses [PMCCFILTR_EL0](#).

Configuration

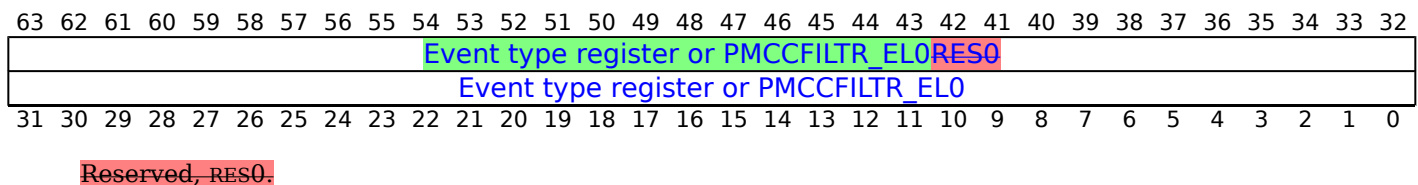
AArch64 System register PMXEVTYPER_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMXEVTYPER\[31:0\]](#).

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMXEVTYPER_EL0 are UNDEFINED.

Attributes

PMXEVTYPER_EL0 is a 64-bit register.

Field descriptions



Bits [31:0]

Bits [63:032]

When [PMSELR_EL0.SEL](#) == 31, this register accesses [PMCCFILTR_EL0](#).

Otherwise, this register accesses [PMEVTYPER<n>_EL0](#) where n is the value in [PMSELR_EL0.SEL](#).

The reset behavior of this field is:

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

Accessing PMXEVTYPER_EL0

If FEAT_FGT is implemented, and [PMSELR_EL0.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_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 not 31 and is greater than or equal to the number of accessible event counters, then reads and writes of [PMXEVTYPER_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 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 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' && (!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 HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMXEVTYPER_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() && (!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 HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMXEVTYPER_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
            return PMXEVTYPER_EL0;
    elsif PSTATE.EL == EL3 then
        return PMXEVTYPER_EL0;

```

MSR PMXEVTYPER_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1101	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' && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') &&
HDFGWTR_EL2.PMEVTYPERn_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
            PMXEVTYPER_EL0 = X[t];
    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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMEVTYPERn_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
            PMXEVTYPER_EL0 = X[t];
    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
            PMXEVTYPER_EL0 = X[t];
    elsif PSTATE.EL == EL3 then
        PMXEVTYPER_EL0 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)htmldiff from-(new)

RNDRRS, Reseeded Random Number

The RNDRRS characteristics are:

Purpose

Reseeded Random Number. Returns a 64-bit random number which is reseeded from the True Random Number source immediately before the read of the random number.

If the hardware returns a genuine random number, PSTATE.NZCV is set to 0b0000.

If the instruction cannot return a genuine random number in a reasonable period of time, PSTATE.NZCV is set to 0b0100 and the data value returned is 0.

When FEAT_RNG_TRAP is implemented and SCR_EL3.TRNDR is 1, reads of this register are trapped to EL3.

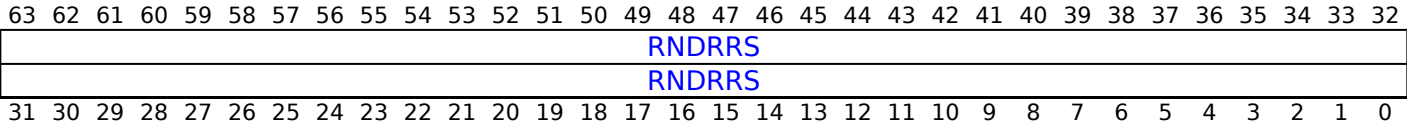
Configuration

This register is present only when FEAT_RNG is implemented or FEAT_RNG_TRAP is implemented. Otherwise, direct accesses to RNDRRS are UNDEFINED.

Attributes

RNDRRS is a 64-bit register.

Field descriptions



RNDRRS, bits [63:0]

Reseeded Random Number. Returns a 64-bit Random Number which is reseeded from the True Random Number source immediately before this read.

The reset behavior of this field is:

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

Accessing RNDRRS

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

MRS <Xt>, RNDRRS

op0	op1	CRn	CRm	op2
0b11	0b011	0b0010	0b0100	0b001


```

if PSTATE.EL == EL0 then
    if IsFeatureImplemented(FEAT_RNG_TRAP) && SCR_EL3.TRNDR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif !IsFeatureImplemented(FEAT_RNG) then
        UNDEFINED;
    else
        return RNDRRS;
elsif PSTATE.EL == EL1 then
    if IsFeatureImplemented(FEAT_RNG_TRAP) && SCR_EL3.TRNDR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif !IsFeatureImplemented(FEAT_RNG) then
        UNDEFINED;
    else
        return RNDRRS;
elsif PSTATE.EL == EL2 then
    if IsFeatureImplemented(FEAT_RNG_TRAP) && SCR_EL3.TRNDR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif !IsFeatureImplemented(FEAT_RNG) then
        UNDEFINED;
    else
        return RNDRRS;
elsif PSTATE.EL == EL3 then
    if IsFeatureImplemented(FEAT_RNG_TRAP) && SCR_EL3.TRNDR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif !IsFeatureImplemented(FEAT_RNG) then
        UNDEFINED;
    else
        return RNDRRS;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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>, IMPLEMENTATION DEFINED maintenance instructions

The SYS S1_<op1>_<Cn>_<Cm>_<op2>, SYSL 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> 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
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.

Executing the SYS S1_<op1>_<Cn>_<Cm>_<op2>, SYSL 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 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
      IMPLEMENTATION_DEFINED "SYS";
elseif PSTATE.EL == EL1 then
  if EL2Enabled() && HCR_EL2.TIDCP == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    IMPLEMENTATION_DEFINED "SYS";
elseif PSTATE.EL == EL2 then
  IMPLEMENTATION_DEFINED "SYS";
elseif PSTATE.EL == EL3 thenelse
  IMPLEMENTATION_DEFINED "SYS";
```

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 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
      IMPLEMENTATION_DEFINED "SYS";
elseif PSTATE.EL == EL1 then
  if EL2Enabled() && HCR_EL2.TIDCP == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  else
    IMPLEMENTATION_DEFINED "SYS";
elseif PSTATE.EL == EL2 then
  IMPLEMENTATION_DEFINED "SYS";
elseif PSTATE.EL == EL3 thenelse
  IMPLEMENTATION_DEFINED "SYS";
```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd5b36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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

There are no configuration notes.

Attributes

S3_<op1>_<Cn>_<Cm>_<op2> is 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 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 SCTL_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' && SCTL_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            IMPLEMENTATION_DEFINED "S3";
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            IMPLEMENTATION_DEFINED "S3";
    elsif PSTATE.EL == EL2 then
        IMPLEMENTATION_DEFINED "S3";
    elsif PSTATE.EL == EL3 then else
        IMPLEMENTATION_DEFINED "S3";

```

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 SCTL_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' && SCTL_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            IMPLEMENTATION_DEFINED "S3";
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            IMPLEMENTATION_DEFINED "S3";
    elsif PSTATE.EL == EL2 then
        IMPLEMENTATION_DEFINED "S3";
    elsif PSTATE.EL == EL3 then else
        IMPLEMENTATION_DEFINED "S3";

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(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

AArch64 System register SCR_EL3 bits [31:0] can be mapped to AArch32 System register [SCR\[31:0\]](#), but this is not architecturally mandated.

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	50	49	48	47	46	45	44	43	42	41	40	
RES0	NSE	RES0														GPF	RES0					EnTP2	TRND	
TWED	EL	TWED	En	ECV	En	FGT	En	ATA	En	SCXT	RES0	FIEN	NMEA	EASE	EEL2	API	APK	TERR	TLOR	TWET	TWI	STRW	SIF	HCE
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	

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 [61:49]

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.

Bits [47:42]

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	<p>This control does not cause any instructions to be trapped and has no effect on reads of RNDRID_AA64ISAR0_EL1 and RDNRRS to be trapped. RNDR.</p> <p>When FEAT_RNG is implemented:</p> <ul style="list-style-type: none"> ID_AA64ISAR0_EL1.RNDR returns the value 0b0001. <p>When FEAT_RNG is not implemented:</p> <ul style="list-style-type: none"> ID_AA64ISAR0_EL1.RNDR returns the value 0b0000. MRS reads of RNDR and RDNRRS are UNDEFINED.
0b1	<p>ID_AA64ISAR0_EL1.RNDR returns the value 0b0001.</p> <p>Any attempt to read RNDR or RNDRRS is trapped to EL3.</p> <p>When FEAT_RNG is not implemented, reads of ID_AA64ISAR0_EL1.RNDR return the value 0b0001.</p>

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 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 [HFGRTR_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 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 $2^{(TWEDEL + 8)}$ 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.

Otherwise:

Reserved, RES0.

ATA, bit [26]

When FEAT_MTE2 is implemented:

Allocation Tag Access. Controls access at EL2, EL1 and EL0 to Allocation Tags.

ATA	Meaning
0b0	Access to Allocation Tags is prevented. Accesses at EL1 and EL2 to GCR_EL1 , RGSER_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 to TFSR_EL12 that are not UNDEFINED are trapped to EL3.
0b1	This control does not prevent access to Allocation Tags.

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:

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](#), [ERXPFPGCTL_EL1](#), and [ERXPFPGF_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. When [SCR_EL3](#).EA == 1, controls whether PSTATE.A masks SError interrupts at EL3.

NMEA	Meaning
0b0	If SCR_EL3 .EA == 1, asserted SError interrupts are not taken at EL3 if PSTATE.A == 1.
0b1	If SCR_EL3 .EA == 1, asserted SError interrupts are taken at EL3 regardless of the value of PSTATE.A.

When SCR_EL3.EA == 0:

- Asserted SError interrupts are not taken at EL3 regardless of the value of PSTATE.A and this field.
- This field is ignored and its Effective value is 0.

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 LDRAB 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 'System register control of 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, PACIZB, 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 'System register control of 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.

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.

When FEAT_WFxT or FEAT_WFxT2 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.

When FEAT_WFxT or FEAT_WFxT2 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.

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]

When FEAT_SEL2 is implemented:

Secure instruction fetch. When the PE is in Secure state, this bit disables instruction fetch from memory marked in the first stage of translation as being Non-secure. **The possible values for this bit are:**

SIF	Meaning
0b0	Secure state instruction fetches from memory marked in the first stage of translation as being Non-secure are permitted.
0b1	Secure state instruction fetches from memory marked in the first stage of translation as being Non-secure are not permitted.

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.

Otherwise:

Secure instruction fetch. When the PE is in Secure state, this bit disables instruction fetch from Non-secure memory.

SIF	Meaning
0b0	Secure state instruction fetches from Non-secure memory are permitted.
0b1	Secure state instruction fetches from Non-secure memory are not permitted.

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.

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.
0b1	Indicates that Exception levels lower than EL3 are in Non-secure state, so memory accesses from those Exception levels cannot access Secure memory.

When SCR_EL3.{EEL2, NS} == {1, 0}, then EL2 is using AArch64 and in Secure state.

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
    return 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];
```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

The SCTRL_EL1 characteristics are:

Purpose

Provides top level control of the system, including its memory system, at EL1 and EL0.

Configuration

AArch64 System register SCTL_R_EL1 bits [31:0] are architecturally mapped to AArch32 System register [SCTL_R\[31:0\]](#).

Attributes

SCTLR_EL1 is a 64-bit register.

Field descriptions

63	62	61	60	59	58	57	56	55	54	53						
TIDCPRES0	SPINTMASK	EnTP2	NMIRES0	EnTP2	EPAN	RES0	EnALS	EPAN	EnAS0	EnALS	EnASR	EnAS0	TME	EnASR	TME0	TMET
EnIA	EnIB	LSMAOE	nTLSMD	EnDA	UCI	EE	E0E	SPAN	EIS	IESB						
31	30	29	28	27	26	25	24	23	22	21						

TIDCP, Bits **bit [63:61]**

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	<p>Instructions accessing the following System register or System instruction spaces are trapped to EL1 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 EL1, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

SPINTMASK, bit [62]**When FEAT_NMI is implemented:**

Superpriority Interrupt Mask enable. When SCTLR_EL1.NMI is 1, controls the value of PSTATE.ALLINT on taking an exception to EL1.

SPINTMASK	Meaning
0b0	PSTATE.ALLINT is set to 1 on taking an exception to EL1.
0b1	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. Enables support for IRQ and FIQ interrupts with Superpriority, and determines additional masking behavior of the PSTATE.I and PSTATE.F flags.

NMI	Meaning
0b0	The behaviour of PSTATE.I and PSTATE.F is unchanged. IRQ and FIQ interrupts with Superpriority have no effect on interrupts that are targeted at EL1.
0b1	IRQ and FIQ interrupts can be marked as having Superpriority as an additional attribute, and additional Superpriority masking behavior is determined by PSTATE.ALLINT.

The reset behavior of this field is:

- 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.

Bits [59:58]

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 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 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 EL1 access to Allocation Tags.

ATA	Meaning
0b0	Access to Allocation Tags is prevented.
0b1	This control does not prevent access to Allocation Tags.

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 EL0 access to Allocation Tags.

ATA0	Meaning
0b0	Access to Allocation Tags is prevented.
0b1	This control does not prevent access to Allocation Tags.

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.	
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 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.	
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 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.

BitBits [34:32]

Reserved, RES0.

MSCEn, bit [33]

When FEAT_MOPS is implemented and (HCR_EL2.E2H == 0 or HCR_EL2.TGE == 0):

MemCpy and MemSet instructions Enable. Enables execution of the MemCpy and MemSet instructions at EL0.

MSCEn	Meaning
0b0	Execution of the MemCpy and MemSet 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 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 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_PAAuth 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_PAuth 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.

The possible values of this bit are:

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.

The possible values of this bit are:

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.

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.

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. ~~The possible values of this bit are:~~

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_WFxFt is implemented, this trap also applies to the WFET instruction.

~~When FEAT_WFxFt or FEAT_WFxFt2 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.

When FEAT_WFxT or FEAT_WFxT2 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, 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 to the following instructions:

- AArch32 CFPRCTX, DVPRCTX and CPPRCTX instructions.
- AArch64 CFP RCTX, DVP RCT and CPP RCTX instructions.

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' in the Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile, section F3.2.5. 0b10100xxxxxxxxxxxx: ADD Rd, PC, #imm 0b01001xxxxxxxxxxxx: 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](#), [HSTCLR](#), 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.

nAA	Meaning
0b0	LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAR, LDARH, LDLAR, LDLARH, STLLR, STLLRH, STLR, STLRH, STLUR, and STLURH generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes for accesses.
0b1	This control bit does not cause LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAR, LDARH, LDLAR, LDLARH, STLLR, STLLRH, STLR, STLRH, STLUR, or STLURH to generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes.

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 value of the [HCR_EL2.DC](#) bit is 1, the PE ignores 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 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 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.

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 value of [HCR_EL2](#).{DC, TGE} 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 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 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() && (!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
        return NVMem[0x110];
    else
        return SCTLR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return SCTLR_EL2;
    else
        return SCTLR_EL1;
elsif PSTATE.EL == EL3 then
    return 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() && (!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];
    else
        SCTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        SCTLR_EL2 = X[t];
    else
        SCTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    SCTLR_EL1 = X[t];

```

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
        return 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
        return SCTLR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        return 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];
    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];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        SCTLR_EL1 = X[t];
    else
        UNDEFINED;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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						
TIDCPRES0	SPINTMASK	EnTP2	NMIRES0	EnTP2	EPAN	RES0	EnALS	EPAN	EnAS0	EnALS	EnASR	EnAS0	TME	EnASR	TME0	TMET
EnIA	EnIB	LSMAOE	nTLSMD	EnDA	UCI	EE	E0E	SPAN	EIS	IESB						
31	30	29	28	27	26	25	24	23	22	21						

TIDCP, Bits bit [63:61]
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:

Superpriority Interrupt Mask enable. When SCTLR_EL2.NMI is 1, controls the value of PSTATE.ALLINT on taking an exception to EL2.

SPINTMASK	Meaning
0b0	PSTATE.ALLINT is set to 1 on taking an exception to EL2.
0b1	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. Enables support for IRQ and FIQ interrupts with Superpriority, and determines additional masking behavior of the PSTATE.I and PSTATE.F flags.

NMI	Meaning
0b0	The behaviour of PSTATE.I and PSTATE.F is unchanged. IRQ and FIQ interrupts with Superpriority have no effect on interrupts that are targeted at EL2.
0b1	IRQ and FIQ interrupts can be marked as having Superpriority as an additional attribute, and additional Superpriority masking behavior is determined by PSTATE.ALLINT.

The 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.

Otherwise:

Reserved, RES0.

Bits [59:58]

Reserved, RES0.

EPAN, bit [57]

When FEAT_PAN3 is implemented, HCR_EL2.E2H == 1 and HCR_EL2.E2HHCR_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.

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.

Otherwise:

Reserved, RES0.

EnAS0, bit [55]

When FEAT_LS64 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.

Otherwise:

Reserved, RES0.

EnASR, bit [54]

When FEAT_LS64 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.

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.

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.

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.

Otherwise:

Reserved, RES0.

TWEDEn, 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.

TWEDEn	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.

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 EL2 access to Allocation Tags.

ATA	Meaning
0b0	Access to Allocation Tags is prevented.
0b1	This control does not prevent access to Allocation Tags.

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 EL0 access to Allocation Tags.

ATA0	Meaning
0b0	Access to Allocation Tags is prevented.
0b1	This control does not prevent access to Allocation Tags.

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.

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.	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.	

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.	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.	

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.

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_BT1 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_BT1 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.

Otherwise:

Reserved, RES0.

Bit [34:32]

Reserved, RES0.

MSCEn, bit [33]

When FEAT_MOPS is implemented, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

MemCpy and MemSet instructions Enable. Enables execution of the MemCpy and MemSet instructions at EL0.

MSCEn	Meaning
0b0	Execution of the MemCpy and MemSet 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.

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_PAuth 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_PAuth 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.

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.

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.

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.

EOE, bit [24]

When HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

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.

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.

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.

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.

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.

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 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.

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.

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_EL0.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.

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 to the following instructions:

- AArch32 CFPRCTX, DVPRCTX and CPPRCTX instructions.
- AArch64 CFP RCTX, DVP RCT and CPP RCTX instructions.

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.

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.

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	Any attempt at EL0 using AArch32 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"> ◦ 0b11xxxxxxxxxxxx: All 32-bit instructions, and the 16-bit instructions B, UDF, SVC, LDM, and STM. ◦ 0b1011xxxxxxxxxxxx: All instructions in 'Miscellaneous 16-bit instructions' in the Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile, section F3.2.5. ◦ 0b10100xxxxxxxxxxx: ADD Rd, PC, #imm ◦ 0b01001xxxxxxxxxxx: LDR Rd, [PC, #imm] ◦ 0b0100x1xxx1111xxx: ADD Rdn, PC; CMP Rn, PC; MOV Rd, PC; BX PC; BLX PC. ◦ 0b010001xx1xxx111: 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.

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.

nAA	Meaning
0b0	LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAR, LDARH, LDLAR, LDLARH, STLLR, STLLRH, STLR, STLRH, STLUR, and STLURH generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes for accesses.
0b1	This control bit does not cause LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAR, LDARH, LDLAR, LDLARH, STLLR, STLLRH, STLR, STLRH, STLUR, or STLURH to generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes.

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 an implementation does not implement CP15BEN, access to this field is **RAO/WI**.

When EL0 can only use AArch64, HCR_EL2.E2H == 1 and HCR_EL2.TGE == 1:

Reserved, RES0.

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.

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
    return SCTLR_EL2;
elsif PSTATE.EL == EL3 then
    return 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];
elsif PSTATE.EL == EL3 then
    SCTLR_EL2 = X[t];

```

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() && (!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
        return NVMem[0x110];
    else
        return SCTLR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return SCTLR_EL2;
    else
        return SCTLR_EL1;
elsif PSTATE.EL == EL3 then
    return 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() && (!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];
    else
        SCTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        SCTLR_EL2 = X[t];
    else
        SCTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    SCTLR_EL1 = X[t];

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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.

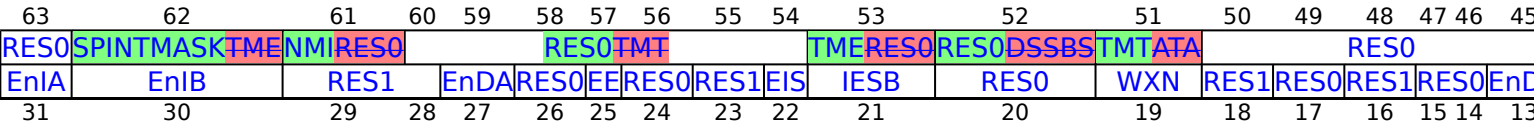
Configuration

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



BitBits [63:54]

Reserved, RES0.

SPINTMASK, bit [62]

When FEAT_NMI is implemented:

Superpriority Interrupt Mask enable. When SCTLR_EL3.NMI is 1, controls the value of PSTATE.ALLINT on taking an exception to EL3.

SPINTMASK	Meaning
0b0	PSTATE.ALLINT is set to 1 on taking an exception to EL3.
0b1	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. Enables support for IRQ and FIQ interrupts with Superpriority, and determines additional masking behavior of the PSTATE.I and PSTATE.F flags.

NMI	Meaning
0b0	The behaviour of PSTATE.I and PSTATE.F is unchanged. IRQ and FIQ interrupts with Superpriority have no effect on interrupts that are targeted at EL3.
0b1	IRQ and FIQ interrupts can be marked as having Superpriority as an additional attribute, and additional Superpriority masking behavior is determined by PSTATE.ALLINT.

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.

Bits [60:54]

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 EL3 access to Allocation Tags.

ATA	Meaning
0b0	Access to Allocation Tags is prevented.
0b1	This control does not prevent access to Allocation Tags.

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_PAuth 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_PAuth 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. ~~The possible values of this bit are:~~

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.

nAA	Meaning
0b0	LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAR, LDARH, LDLAR, LDLARH, STLLR, STLLRH, STLR, STLRH, STLUR, and STLURH generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes for accesses.
0b1	This control bit does not cause LDAPR, LDAPRH, LDAPUR, LDAPURH, LDAPURSH, LDAPURSW, LDAR, LDARH, LDLAR, LDLARH, STLLR, STLLRH, STLR, STLRH, STLUR, or STLURH to generate an Alignment fault if all bytes being accessed are not within a single 16-byte quantity, aligned to 16 bytes.

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.

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
    return 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];
```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The SMCR_EL1 characteristics are:

Purpose

This register controls aspects of Streaming SVE that are visible at Exception levels EL1 and EL0.

Configuration

This register is present only when FEAT_SME is implemented. Otherwise, direct accesses to SMCR_EL1 are UNDEFINED.

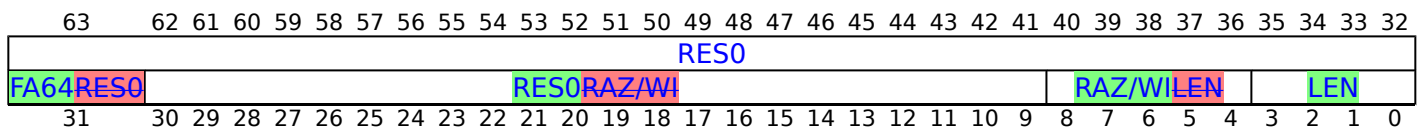
This register has no effect if the PE is not in Streaming SVE mode.

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](#) and [EL1](#). [EL0](#).

Attributes

SMCR_EL1 is a 64-bit register.

Field descriptions



Bits [63:329]

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 EL1 and EL0, if they are treated as legal at more privileged Exception levels in the current Security state.

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.

Bits [30:9]

Reserved, RES0.

Bits [8:4]

Reserved, RAZ/WI.

LEN, bits [3:0]

Effective Streaming SVE Vector Length (SVL).

Constrains the effective Streaming SVE vector register length for EL1 and EL0 to $(LEN+1)*128$ bits. SVL bits only takes effect when the PE is in Streaming SVE mode.

An implementation is permitted to include any set of Streaming SVE vector lengths that are powers of two, from 128 bits to 2048 bits inclusive.

For all purposes other than returning the result of a direct read of SMCR_EL1, this field selects the effective vector length as follows:

- If the requested length is smaller than the minimum implemented Streaming SVE vector length, then the minimum implemented Streaming SVE vector length is used.
- If the requested length is larger than the effective vector length at the next more privileged Exception level in the current Security state, if any, then the effective vector length at the more privileged Exception level is used.
- If the requested length is not implemented, then the requested length rounded down to the nearest implemented Streaming SVE vector length is used.
- Otherwise, the requested length is used.

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);
        endif
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x1F0];
    else
        return 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
        return SMCR_EL2;
    else
        return SMCR_EL1;
endif
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        return SMCR_EL1;
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];
        else
            SMCR_EL1 = X[t];
    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];
        else
            SMCR_EL1 = X[t];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.ESM == '0' then
            AArch64.SystemAccessTrap(EL3, 0x1D);
        else
            SMCR_EL1 = X[t];

```

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
        return 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
                return 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
            return 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];
    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
            SMCR_EL1 = X[t];
    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];
    else
        UNDEFINED;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbfdb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The SMCR_EL2 characteristics are:

Purpose

This register controls aspects of Streaming SVE that are visible at Exception levels EL2, EL1, and EL0.

Configuration

This register is present only when FEAT_SME is implemented. Otherwise, direct accesses to SMCR_EL2 are UNDEFINED.

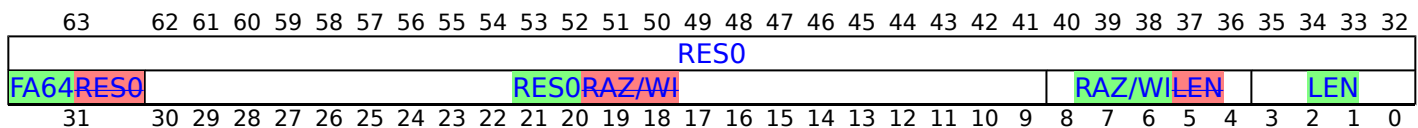
This register has no effect if the PE is not in Streaming SVE mode, or if EL2 is not enabled in the current Security state.

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

Attributes

SMCR_EL2 is a 64-bit register.

Field descriptions



Bits [63:329]

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 EL2, if they are treated as legal 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.

Bits [30:9]

Reserved, RES0.

Bits [8:4]

Reserved, RAZ/WI.

LEN, bits [3:0]

Effective Streaming SVE Vector Length (SVL).

Constrains the effective Streaming SVE vector register length for EL2, EL1, and EL0 to (LEN+1)*128 bits when EL2 is enabled in the current Security state. SVL state only and takes effect when the PE is in Streaming SVE mode.

An implementation is permitted to include any set of Streaming SVE vector lengths that are powers of two, from 128 bits to 2048 bits inclusive.

For all purposes other than returning the result of a direct read of SMCR_EL2, this field selects the effective vector length as follows:

- If the requested length is smaller than the minimum implemented Streaming SVE vector length, then the minimum implemented Streaming SVE vector length is used.
- If the requested length is larger than the effective vector length at the next more privileged Exception level in the current Security state, if any, then the effective vector length at the more privileged Exception level is used.
- If the requested length is not implemented, then the requested length rounded down to the nearest implemented Streaming SVE vector length is used.
- Otherwise, the requested length is used.

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
        return SMCR_EL2;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        return 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];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        SMCR_EL2 = X[t];

```

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
            return NVMem[0x1F0];
        else
            return 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
                return SMCR_EL2;
            else
                return SMCR_EL1;
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.ESM == '0' then
            AArch64.SystemAccessTrap(EL3, 0x1D);
        else
            return 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);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
            NVMem[0x1F0] = X[t];
        else
            SMCR_EL1 = X[t];
    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];
        else
            SMCR_EL1 = X[t];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.ESM == '0' then
            AArch64.SystemAccessTrap(EL3, 0x1D);
        else
            SMCR_EL1 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd36e47856e443a7ce9a85f5e501ca

Copyright © 2010-2021 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

Diagram illustrating the structure of the 64-bit RAZ/WI field. The field is divided into three main sections:

- FA64RES0** (bits 63-31): A 33-bit field.
- RES0RAZ/WI** (bits 30-17): A 14-bit field, further divided into:
 - RES0** (bits 30-18): A 13-bit field.
 - RAZ/WI** (bits 17-16): A 2-bit field.
- RAZ/WILEN** (bits 16-0): A 17-bit field, further divided into:
 - RAZ/WI** (bits 16-5): A 12-bit field.
 - LEN** (bits 4-0): A 5-bit field.

Bits [63:329]

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.

Bits [30:9]

Reserved, RES0.

Bits [8:4]

Reserved, RAZ/WI.

LEN, bits [3:0]

Effective Streaming SVE Vector Length (SVL).

Constrains the effective Streaming SVE vector register length for all Exception levels to (LEN+1)*128 bits. SVLbits only takes effect when the PE is in Streaming SVE mode.

An implementation is permitted to include any set of Streaming SVE vector lengths that are powers of two, from 128 bits to 2048 bits inclusive.

For all purposes other than returning the result of a direct read of SMCR_EL3, this field selects the effective vector length as follows:

- If the requested length is smaller than the minimum implemented Streaming SVE vector length, then the minimum implemented Streaming SVE vector length is used.
- If the requested length is not implemented, then the requested length rounded down to the nearest implemented Streaming SVE vector length is used.
- Otherwise, the requested length is used.

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.

Accessing SMCR_EL3

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

MRS <Xt>, SMCR_EL3

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0010	0b110

```

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.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        return SMCR_EL3;
    end
end
    
```

MSR SMCR_EL3, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b110	0b0001	0b0010	0b110

```
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.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        SMCR_EL3 = X[t];
```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

(old)

htmldiff from-

(new)

SMIDR_EL1, Streaming Mode Identification Register

The SMIDR_EL1 characteristics are:

Purpose

Provides additional identification mechanisms for scheduling purposes, for a PE that supports Streaming SVE mode.

Configuration

This register is present only when FEAT_SME is implemented. Otherwise, direct accesses to SMIDR_EL1 are UNDEFINED.

Attributes

SMIDR_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
Implementer																Affinity															
Revision																RES0															
SMPS																RES0															

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.

It is not required that this value is the same as the value of [MIDR_EL1.Implementer](#).

Access to this field is **RO**.

Revision, bits [23:16]

Revision number for the Streaming Mode Compute Unit (SMCU).

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

SMPS, bit [15]

Indicates support for Streaming SVE mode execution priority.

SMPS	Meaning
0b0	Priority control not supported.
0b1	Priority control supported.

Bits [14:12]

Reserved, RES0.

Affinity, bits [11:0]

The SMCU affinity of the accessing PE.

- A value of zero indicates that the PE's implementation of Streaming SVE mode is not shared with other PEs.
- Otherwise, the value identifies which SMCU is associated with this PE. The Affinity value associated with each SMCU is unique within the system as a whole.

Accessing SMIDR_EL1

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

MRS <Xt>, SMIDR_EL1

op0	op1	CRn	CRm	op2
0b11	0b001	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;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return SMIDR_EL1;
elseif PSTATE.EL == EL2 then
    return SMIDR_EL1;
elseif PSTATE.EL == EL3 then
    return SMIDR_EL1;

```

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

(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																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() && (!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
        return 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
        return SMPRI_EL1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return 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() && (!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];
    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];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.ESM == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            SMPRI_EL1 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

SMPRIMAP_EL2, Streaming Mode Priority Mapping Register

The SMPRIMAP_EL2 characteristics are:

Purpose

Maps the value in [SMPRI_EL1](#) to a streaming execution priority value for instructions executed at EL1 and EL0 in the same Security states as EL2.

Configuration

This register is present only when FEAT_SME is implemented. Otherwise, direct accesses to SMPRIMAP_EL2 are UNDEFINED.

When [SMIDR_EL1](#).SMPS is '0', this register is RES0.

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

Attributes

SMPRIMAP_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
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

When all of the following are true, the value in [SMPRI_EL1](#) is mapped to a streaming execution priority using this register:

- The current Exception level is EL1 or EL0.
- EL2 is implemented and enabled in the current Security state.
- [HCRX_EL2](#).SMPME is '1'.

Otherwise, [SMPRI_EL1](#) holds the streaming execution priority value.

P15, bits [63:60]

Priority Mapping Entry 15. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#).Priority value is '15'.

This value is the highest streaming execution priority.

The reset behavior of this field is:

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

P14, bits [59:56]

Priority Mapping Entry 14. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#).Priority value is '14'.

The reset behavior of this field is:

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

P13, bits [55:52]

Priority Mapping Entry 13. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '13'.

The reset behavior of this field is:

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

P12, bits [51:48]

Priority Mapping Entry 12. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '12'.

The reset behavior of this field is:

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

P11, bits [47:44]

Priority Mapping Entry 11. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '11'.

The reset behavior of this field is:

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

P10, bits [43:40]

Priority Mapping Entry 10. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '10'.

The reset behavior of this field is:

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

P9, bits [39:36]

Priority Mapping Entry 9. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '9'.

The reset behavior of this field is:

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

P8, bits [35:32]

Priority Mapping Entry 8. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '8'.

The reset behavior of this field is:

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

P7, bits [31:28]

Priority Mapping Entry 7. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '7'.

The reset behavior of this field is:

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

P6, bits [27:24]

Priority Mapping Entry 6. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '6'.

The reset behavior of this field is:

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

P5, bits [23:20]

Priority Mapping Entry 5. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '5'.

The reset behavior of this field is:

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

P4, bits [19:16]

Priority Mapping Entry 4. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '4'.

The reset behavior of this field is:

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

P3, bits [15:12]

Priority Mapping Entry 3. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '3'.

The reset behavior of this field is:

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

P2, bits [11:8]

Priority Mapping Entry 2. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '2'.

The reset behavior of this field is:

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

P1, bits [7:4]

Priority Mapping Entry 1. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '1'.

The reset behavior of this field is:

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

P0, bits [3:0]

Priority Mapping Entry 0. This entry is used when priority mapping is supported and enabled, and the [SMPRI_EL1](#). Priority value is '0'.

This value is the lowest streaming execution priority.

The reset behavior of this field is:

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

Accessing SMPRIMAP_EL2

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

MRS <Xt>, SMPRIMAP_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x1E8];
    elsif EL2Enabled() && HCR_EL2.NV == '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 HaveEL(EL3) && CPTR_EL3.ESM == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            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 HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return SMPRIMAP_EL2;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return SMPRIMAP_EL2;

```

MSR SMPRIMAP_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x1E8] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '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 HaveEL(EL3) && CPTR_EL3.ESM == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                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 HaveEL(EL3) && CPTR_EL3.ESM == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            SMPRIMAP_EL2 = X[t];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.ESM == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            SMPRIMAP_EL2 = X[t];

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

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.

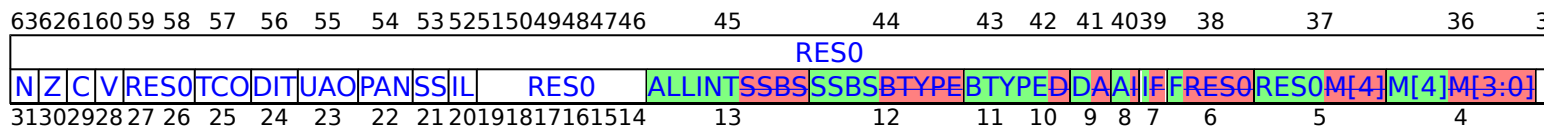
M[3:0]	Meaning
0b0000	User.
0b0001	FIQ.
0b0010	IRQ.
0b0011	Supervisor.
0b0111	Abort.
0b1011	Undefined.
0b1111	System.

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:



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.

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_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] is set to the value of PSTATE.EL on taking an exception to EL1 and copied to PSTATE.EL on executing an 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
        return NVMem[0x160];
    else
        return SPSR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return SPSR_EL2;
    else
        return SPSR_EL1;
elsif PSTATE.EL == EL3 then
    return 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];
    else
        SPSR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        SPSR_EL2 = X[t];
    else
        SPSR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    SPSR_EL1 = X[t];

```


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
        return 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
        return SPSR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        return 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];
    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];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        SPSR_EL1 = X[t];
    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
        return SPSR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return SPSR_EL2;
elsif PSTATE.EL == EL3 then
    return 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];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    SPSR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    SPSR_EL2 = X[t];

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 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]	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 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.

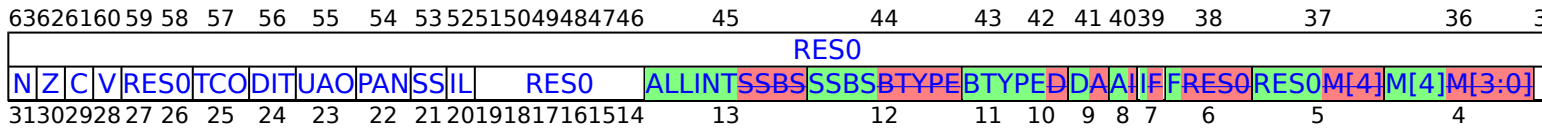
M[3:0]	Meaning
0b0000	User.
0b0001	FIQ.
0b0010	IRQ.
0b0011	Supervisor.
0b0111	Abort.
0b1010	Hyp.
0b1011	Undefined.
0b1111	System.

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 [63:32]

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.

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_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 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 [19:14:13]

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]

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.

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
        return SPSR_EL1;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return SPSR_EL2;
elsif PSTATE.EL == EL3 then
    return 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];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    SPSR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    SPSR_EL2 = X[t];

```

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
        return NVMem[0x160];
    else
        return SPSR_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return SPSR_EL2;
    else
        return SPSR_EL1;
elsif PSTATE.EL == EL3 then
    return 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];
    else
        SPSR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        SPSR_EL2 = X[t];
    else
        SPSR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    SPSR_EL1 = X[t];

```

(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

AArch64 System register SPSR_EL3 bits [31:0] can be mapped to AArch32 System register [SPSR_mon\[31:0\]](#), but this is not architecturally mandated.

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.

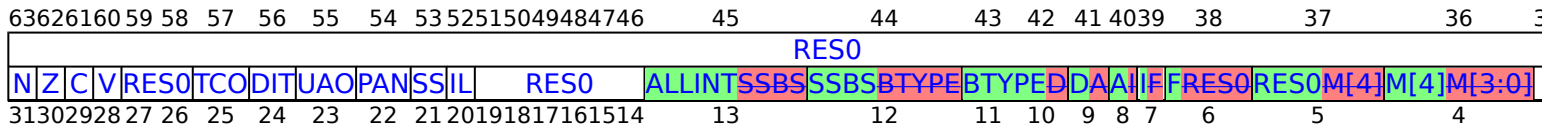
M[3:0]	Meaning
0b0000	User.
0b0001	FIQ.
0b0010	IRQ.
0b0011	Supervisor.
0b0110	Monitor.
0b0111	Abort.
0b1010	Hyp.
0b1011	Undefined.
0b1111	System.

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:



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.

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_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 [19:14:13]

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]

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.

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
    return 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];
```

(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 ZA array storage. The possible values of this bit are:

ZA	Meaning
0b0	SME ZA array storage is invalid and not accessible. SME instructions that access the ZA array are illegal.
0b1	SME ZA array storage is valid and accessible. If SME instructions are not trapped, SME instructions that access the ZA array are legal.

When the PSTATE.ZA value is of changed PSTATE.ZA by is any changed means from 0 to 1, the following contents applies: of the SME ZA storage are set to zero.

When PSTATE.ZA is changed by any means from 1 to 0, there is no observable change to SME ZA storage.

When PSTATE.ZA is changed by any means from 1 to 0 or from 0 to 1, there is no effect on the Streaming SVE vector and predicate registers and FPSR if PSTATE.SM remains set to 1.

- When changed from 0 to 1, all implemented bits of the SME ZA storage are set to zero.
- When changed from 1 to 0, there is no observable change to SME ZA storage.
- There is no 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. 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.

- The value that bit held before the PE entered Streaming SVE mode, but only if that bit is also implemented when the PE is not in Streaming SVE mode.
- The value zero.

When the effective value of PSTATE.SM is changed by any means from 1 to 0, an exit from Streaming SVE mode is performed, and each implemented bit of the SVE registers Z0-Z31 and P0-P15 in the new mode is set to an IMPLEMENTATION DEFINED choice of one of the following values:

- The value that bit held before the PE exited Streaming SVE mode, but only if that bit is also implemented when the PE is in Streaming SVE mode.
- The value zero.

When the effective value of PSTATE.SM is changed by any means from 1 to 0, each implemented bit of the FFR predicate register in the new mode is set to zero.

When the effective value of PSTATE.SM is changed by any means from 0 to 1 or from 1 to 0, the FPSR in the new mode is set to an IMPLEMENTATION DEFINED choice of one of the following values:

When the value of PSTATE.SM is changed, the following applies:

When the effective value of PSTATE.SM is changed by any means from 0 to 1, an entry to Streaming SVE mode is performed, and each implemented bit of the SVE registers Z0-Z31 and P0-P15 in the new mode is set to an IMPLEMENTATION DEFINED choice of one of the following values:

- When the changed value from that 0 to 1, an entry to Streaming SVE mode is performed, the FPSR held in the previous mode, before PSTATE.SM was changed.
- When the changed value from 0x0000_0000_0800_009f, 1 in to which 0, all and of exit the from cumulative Streaming status bits SVE are modeset is to performed. 1.
- 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.
- There is no effect on ZA 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
        return 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
        return 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
        return Zeros(62):PSTATE.<ZA,SM>;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        return 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;
    elseif !(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);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.SMEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elseif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elseif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        bits(64)PSTATE.<ZA,SM> v = X[t]; SetPSTATE_SM(v<1:0>); SetPSTATE_ZA(v<1>);>;
elseif 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;
    elseif CPACR_EL1.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x1D);
    elseif EL2Enabled() && HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elseif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        bits(64)PSTATE.<ZA,SM> v = X[t]; SetPSTATE_SM(v<1:0>); SetPSTATE_ZA(v<1>);>;
elseif 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;
    elseif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elseif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elseif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        bits(64)PSTATE.<ZA,SM> v = X[t]; SetPSTATE_SM(v<1:0>); SetPSTATE_ZA(v<1>);>;
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        bits(64)PSTATE.<ZA,SM> v = X[t]; SetPSTATE_SM(v<1:0>); SetPSTATE_ZA(v<1>);>;

```

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

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

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

The TLBI PAALL characteristics are:

Purpose

Invalidates cached copies of GPT entries from TLBs. Details:

- The invalidation applies to TLB entries containing GPT information **that relates** to a physical address.
- The invalidation applies to all TLB entries containing GPT information.
- The invalidation affects only the TLBs for the PE executing the operation.

The full set of TLB maintenance instructions that invalidate cached GPT entries is: [TLBI PAALL](#), [TLBI PAALLOS](#), [TLBI RPALOS](#), and [TLBI RPAOS](#).

These instructions have the same ordering, observability, and completion behavior as all other TLBI instructions.

Configuration

This instruction is present only when FEAT_RME is implemented. Otherwise, direct accesses to TLBI PAALL are UNDEFINED.

Attributes

TLBI PAALL 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 PAALL instruction

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

TLBI PAALL{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0111	0b100

```

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_PAALL(Shareability_NSH);

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

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

The TLBI PAALLOS characteristics are:

Purpose

Invalidates cached copies of GPT entries from TLBs. Details:

- The invalidation applies to TLB entries containing GPT information **that relates** to a physical address.
- The invalidation applies to all TLB entries containing GPT information.
- The invalidation affects all TLBs in the Outer Shareable domain.

The full set of TLB maintenance instructions that invalidate cached GPT entries is: [TLBI PAALL](#), [TLBI PAALLOS](#), [TLBI RPALOS](#), and [TLBI RPAOS](#).

These instructions have the same ordering, observability, and completion behavior as all other TLBI instructions.

Configuration

This instruction is present only when FEAT_RME is implemented. Otherwise, direct accesses to TLBI PAALLOS are UNDEFINED.

Attributes

TLBI PAALLOS 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 PAALLOS instruction

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

TLBI PAALLOS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0001	0b100

```

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_PAALL(Shareability_OSH);

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

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

The TLBI RPALOS characteristics are:

Purpose

Invalidates cached copies of GPT entries from TLBs. Details:

- The invalidation applies to TLB entries containing GPT information **that relates** to a physical address.
- The invalidation affects all TLBs in the Outer Shareable domain.
- Invalidates TLB entries containing GPT information from the **final** level of the GPT walk that relates to the supplied physical address.
- Invalidations are range-based, invalidating TLB entries starting from the address in BaseADDR, within the range as specified by SIZE.

The full set of TLB maintenance instructions that invalidate cached GPT entries is: [TLBI PAALL](#), [TLBI PAALLOS](#), [TLBI RPALOS](#), and [TLBI RPAOS](#).

These instructions have the same ordering, observability, and completion behavior as all other TLBI instructions.

Configuration

This instruction is present only when FEAT_RME is implemented. Otherwise, direct accesses to TLBI RPALOS are UNDEFINED.

Attributes

TLBI RPALOS 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																SIZE				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				

Bits [63:48]

Reserved, RES0.

SIZE, bits [47:44]

Size of the range for invalidation.

If SIZE is a reserved value, no TLB entries are required to be invalidated.

SIZE	Meaning
0b0000	4KB.
0b0001	16KB.
0b0010	64KB.
0b0011	2MB.
0b0100	32MB.
0b0101	512MB.
0b0110	1GB.
0b0111	16GB.
0b1000	64GB.
0b1001	512GB.

All other values are reserved.

If SIZE gives a range smaller than the configured physical granule size in [GPCCR_EL3.PGS](#), then the effective value of SIZE is taken to be the size configured by [GPCCR_EL3.PGS](#).

If [GPCCR_EL3.PGS](#) is configured to a reserved value, no TLB entries are required to be invalidated.

Bits [43:40]

Reserved, RES0.

Address, bits [39:0]

The starting address for the range of the maintenance instruction.

This field is decoded with reference to the value of [GPCCR_EL3.PGS](#) to give BaseADDR as follows:

GPCCR_EL3.PGS	BaseADDR
0b00 (4KB)	BaseADDR[51:12] = Xt[39:0]
0b10 (16KB)	BaseADDR[51:14] = Xt[39:2]
0b01 (64KB)	BaseADDR[51:16] = Xt[39:4]

Other bits of BaseADDR are treated as zero, to give the effective value of BaseADDR.

If the effective value of BaseADDR is not aligned to the size of the effective value of SIZE, no TLB entries are required to be invalidated.

If [GPCCR_EL3.PGS](#) is configured to a reserved value, no TLB entries are required to be invalidated.

Executing the TLBI RPALOS instruction

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

TLBI RPALOS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0100	0b111

```

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_RPA(TLBILevel_Last, X[t], Shareability_0SH);

```

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

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

The TLBI RPAOS characteristics are:

Purpose

Invalidates cached copies of GPT entries from TLBs. Details:

- The invalidation applies to TLB entries containing GPT information **that relates** to a physical address.
- The invalidation affects all TLBs in the Outer Shareable domain.
- Invalidates TLB entries containing GPT information from all levels of the GPT walk that relates to the supplied physical address.
- Invalidations are range-based, invalidating TLB entries starting from the address in BaseADDR, within the range as specified by SIZE.

The full set of TLB maintenance instructions that invalidate cached GPT entries is: [TLBI PAALL](#), [TLBI PAALLOS](#), [TLBI RPALOS](#), and [TLBI RPAOS](#).

These instructions have the same ordering, observability, and completion behavior as all other TLBI instructions.

Configuration

This instruction is present only when FEAT_RME is implemented. Otherwise, direct accesses to TLBI RPAOS are UNDEFINED.

Attributes

TLBI RPAOS 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																SIZE				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

Bits [63:48]

Reserved, RES0.

SIZE, bits [47:44]

Size of the range for invalidation.

If SIZE is a reserved value, no TLB entries are required to be invalidated.

SIZE	Meaning
0b0000	4KB.
0b0001	16KB.
0b0010	64KB.
0b0011	2MB.
0b0100	32MB.
0b0101	512MB.
0b0110	1GB.
0b0111	16GB.
0b1000	64GB.
0b1001	512GB.

All other values are reserved.

If SIZE gives a range smaller than the configured physical granule size in [GPCCR_EL3.PGS](#), then the effective value of SIZE is taken to be the size configured by [GPCCR_EL3.PGS](#).

If [GPCCR_EL3.PGS](#) is configured to a reserved value, no TLB entries are required to be invalidated.

Bits [43:40]

Reserved, RES0.

Address, bits [39:0]

The starting address for the range of the maintenance instruction.

This field is decoded with reference to the value of [GPCCR_EL3.PGS](#) to give BaseADDR as follows:

GPCCR_EL3.PGS	BaseADDR
0b00 (4KB)	BaseADDR[51:12] = Xt[39:0]
0b10 (16KB)	BaseADDR[51:14] = Xt[39:2]
0b01 (64KB)	BaseADDR[51:16] = Xt[39:4]

Other bits of BaseADDR are treated as zero, to give the effective value of BaseADDR.

If the effective value of BaseADDR is not aligned to the size of the effective value of SIZE, no TLB entries are required to be invalidated.

If [GPCCR_EL3.PGS](#) is configured to a reserved value, no TLB entries are required to be invalidated.

Executing the TLBI RPAOS instruction

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

TLBI RPAOS{, <Xt>}

op0	op1	CRn	CRm	op2
0b01	0b110	0b1000	0b0100	0b011

```

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_RPA(TLBILevel_Any, X[t], Shareability_0SH);

```

(old)

htmldiff from-

(new)

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.

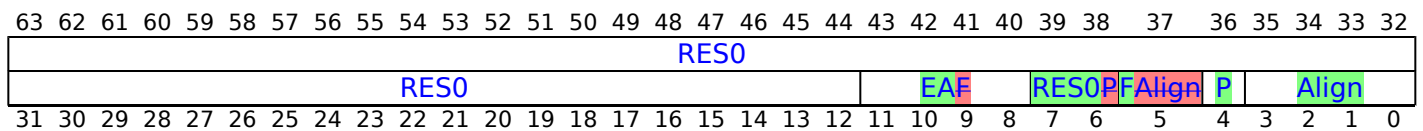
Configuration

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



Bits [63:126]

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.

Access to this field is **RO**.

Otherwise:

Reserved, RES0.

Bits [7:6]

Reserved, RES0.

F, bit [5]

Flag **updates**. **Updates**. **Describes** **Defines** how whether the address **translation** **translation** performed by the Trace Buffer Unit **manages** the Access **flag** **Flag** and dirty state. **Defined values are:**

F	Meaning
0b0	Hardware Trace management buffer of address translation does not manage the Access flag and dirty state for in accesses made by the Trace Buffer Unit is always disabled for all translation stages . tables .
0b1	Hardware Trace management buffer of address translation manages the Access flag 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 MMU buffer on owning this translation regime . PE .

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.

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.

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;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.TRBIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return TRBIDR_EL1;
elseif PSTATE.EL == EL2 then
    return TRBIDR_EL1;
elseif PSTATE.EL == EL3 then
    return TRBIDR_EL1;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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	32
RES0																															
EC				RES0				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 [63:32]

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, other than GPF, on write to trace buffer.	MSS encoding for Granule Protection Check fault MSS encoding for other trace buffer management events	When FEAT_RME is implemented
0b011111	Buffer management event for 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 [25:23]

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]

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**.

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]

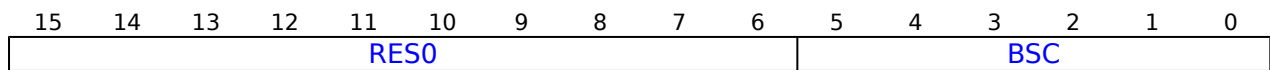
Management Event Specific Syndrome. Contains syndrome specific to the management event.

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
0b000000	Collection not stopped.
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.

All other values are reserved.

MSS encoding for Buffer management event for IMPLEMENTATION DEFINED reason



IMPLEMENTATION DEFINED, bits [15:0]

IMPLEMENTATION DEFINED.

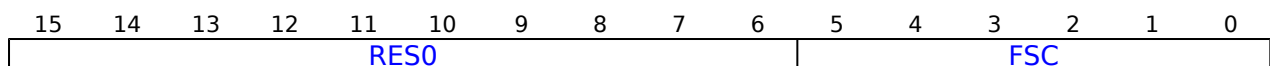
MSS encoding for Granule Protection Check fault



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.	
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.	
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.	When FEAT_HAFDBS is implemented
0b110001	Unsupported atomic hardware update fault.	

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() && (!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
            return 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
            return TRBSR_EL1;
elsif PSTATE.EL == EL3 then
    return 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() && (!HaveEL(EL3) || SCR_EL3.FGTEn == '1') && HDBGWTR_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];
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];
elsif PSTATE.EL == EL3 then
    TRBSR_EL1 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TTBR0_EL1, Translation Table Base Register 0 (EL1)

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\[63:0\]](#).

Attributes

TTBR0_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
ASID																BADDR[47:1]															
																BADDR[47:1]															
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

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.

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 **bitfield** 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() && (!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
        return NVMem[0x200];
    else
        return TTBR0_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return TTBR0_EL2;
    else
        return TTBR0_EL1;
elsif PSTATE.EL == EL3 then
    return TTBR0_EL1;

```

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() && (!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];
    else
        TTBR0_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TTBR0_EL2 = X[t];
    else
        TTBR0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    TTBR0_EL1 = X[t];

```

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
        return 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
        return TTBR0_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        return TTBR0_EL1;
    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];
    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 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        TTBR0_EL1 = X[t];
    else
        UNDEFINED;

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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.

Attributes

TTBR0_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
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

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.

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 **bitfield** 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
    return TTBR0_EL2;
elsif PSTATE.EL == EL3 then
    return TTBR0_EL2;

```

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 = X[t];
elsif PSTATE.EL == EL3 then
    TTBR0_EL2 = X[t];

```

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() && (!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
        return NVMem[0x200];
    else
        return TTBR0_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return TTBR0_EL2;
    else
        return TTBR0_EL1;
elsif PSTATE.EL == EL3 then
    return TTBR0_EL1;

```

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() && (!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];
    else
        TTBR0_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TTBR0_EL2 = X[t];
    else
        TTBR0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    TTBR0_EL1 = X[t];

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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

63	62	61	60	59	58	57	56	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[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 [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.

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 **bitfield** 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
    return 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];
```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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\[63:0\]](#).

Attributes

TTBR1_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
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

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.

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 **bitfield** 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() && (!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
        return NVMem[0x210];
    else
        return TTBR1_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return TTBR1_EL2;
    else
        return TTBR1_EL1;
elsif PSTATE.EL == EL3 then
    return TTBR1_EL1;

```

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() && (!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];
    else
        TTBR1_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TTBR1_EL2 = X[t];
    else
        TTBR1_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    TTBR1_EL1 = X[t];

```

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
        return 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
        return TTBR1_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        return TTBR1_EL1;
    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];
    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 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' then
        TTBR1_EL1 = X[t];
    else
        UNDEFINED;

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbf36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

TTBR1_EL2, Translation Table Base Register 1 (EL2)

The TTBR1_EL2 characteristics are:

Purpose

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 higher VA range in the EL2&0 translation regime, and other information for this translation regime.

Note

When [HCR_EL2.E2H](#) is 0, the contents of this register are ignored by the PE, except for a direct read or write of the register.

Configuration

This register is present only when FEAT_VHE is implemented. Otherwise, direct accesses to TTBR1_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

TTBR1_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	
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	

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.

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 **bitfield** 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
    return TTBR1_EL2;
elsif PSTATE.EL == EL3 then
    return TTBR1_EL2;

```

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 = X[t];
elsif PSTATE.EL == EL3 then
    TTBR1_EL2 = X[t];

```

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() && (!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
        return NVMem[0x210];
    else
        return TTBR1_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return TTBR1_EL2;
    else
        return TTBR1_EL1;
elsif PSTATE.EL == EL3 then
    return TTBR1_EL1;

```

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() && (!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];
    else
        TTBR1_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        TTBR1_EL2 = X[t];
    else
        TTBR1_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    TTBR1_EL1 = X[t];

```


3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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

63	62	61	60	59	58	57	56	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																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 [63:48]

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 1 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 1 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 1 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.

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 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 **bitfield** 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 !IsSecure() then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x030];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsSecure() then
        UNDEFINED;
    else
        return VSTTBR_EL2;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        return 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 !IsSecure() then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x030] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsSecure() then
        UNDEFINED;
    else
        VSTTBR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        VSTTBR_EL2 = X[t];

```

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

VTTBR_EL2, Virtualization Translation Table Base Register

The VTTBR_EL2 characteristics are:

Purpose

Holds the base address of the translation table for the initial lookup for stage 2 of an address translation in the EL1&0 translation regime, and other information for this translation regime.

Configuration

AArch64 System register VTTBR_EL2 bits [63:0] are architecturally mapped to AArch32 System register [VTTBR\[63:0\]](#).

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

VTTBR_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	
VMID																BADDR																
																BADDR																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	

VMID, bits [63:48]

VMID encoding when FEAT_VMID16 is implemented and (VTCR_EL2.VS == 1 or AArch32 is supported)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VMID															

VMID, bits [15:0]

The VMID for the translation table.

If ~~EL2 is using AArch32, or 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 or the implementation only supports execution in AArch64 state)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RES0								VMID							

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.
- The [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 1 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 1 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 1 translation table base address are zero.
- In an implementation that includes FEAT_TTCNP, bit[0] of the stage 1 translation table base address is 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 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.

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.

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 **bitfield** 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
        return NVMem[0x020];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return VTTBR_EL2;
elsif PSTATE.EL == EL3 then
    return VTTBR_EL2;

```

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];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    VTTBR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    VTTBR_EL2 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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 if 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															
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]

Effective SVE Vector Length (VL).

Constrains the effective scalable vector register length for EL1 and EL0 to (LEN+1)*128 **bits. VL only takes effect** when the PE is not in Streaming SVE mode.

An implementation is permitted to include any set of vector lengths that are multiples of 128 bits, from 128 bits to 2048 bits inclusive, and required to support all vector lengths that are powers of two, from 128 bits up to its maximum implemented vector length.

For all purposes other than returning the result of a direct read of ZCR_EL1, this field selects the effective vector length as follows:

- If the requested length is larger than the effective vector length at the next more privileged Exception level in the current Security state, if any, then the effective vector length at the more privileged Exception level is used.
- If the requested length is not implemented, then the requested length rounded down to the nearest implemented scalable vector length is used.

- Otherwise, the requested length is used.

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
        return NVMem[0x1E0];
    else
        return 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
        return ZCR_EL2;
    else
        return ZCR_EL1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        return 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];
    else
        ZCR_EL1 = X[t];
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];
    else
        ZCR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        ZCR_EL1 = X[t];

```

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
        return 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
            return 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
            return 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];
    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];
    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];
    else
        UNDEFINED;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbfdb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(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 if EL2 is not enabled in the current Security state, or if 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]

Effective SVE Vector Length (VL).

Constrains the effective scalable vector register length for EL2, EL1, and EL0 to (LEN+1)*128 bits when EL2 is enabled in the current Security state. VL state only takes effect and when the PE is not in Streaming SVE mode.

An implementation is permitted to include any set of vector lengths that are multiples of 128 bits, from 128 bits to 2048 bits inclusive, and required to support all vector lengths that are powers of two, from 128 bits up to its maximum implemented vector length.

For all purposes other than returning the result of a direct read of ZCR_EL2, this field selects the effective vector length as follows:

- If the requested length is larger than the effective vector length at the next more privileged Exception level in the current Security state, if any, then the effective vector length at the more privileged Exception level is used.
- If the requested length is not implemented, then the requested length rounded down to the nearest implemented scalable vector length is used.
- Otherwise, the requested length is used.

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
        return ZCR_EL2;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        return 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];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        ZCR_EL2 = X[t];

```

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
        return NVMem[0x1E0];
    else
        return 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
        return ZCR_EL2;
    else
        return ZCR_EL1;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        return 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];
    else
        ZCR_EL1 = X[t];
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];
    else
        ZCR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        ZCR_EL1 = X[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd36e47856e443a7ce9a85f5e501ca

Copyright © 2010-2021 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 if 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						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]

Effective SVE Vector Length (VL).

Constrains the effective scalable vector register length for all Exception levels to (LEN+1)*128 **bits. VL only takes effect**bits when the PE is not in Streaming SVE mode.

An implementation is permitted to include any set of vector lengths that are multiples of 128 bits, from 128 bits to 2048 bits inclusive, and required to support all vector lengths that are powers of two, from 128 bits up to its maximum implemented vector length.

For all purposes other than returning the result of a direct read of ZCR_EL3, this field selects the effective vector length as follows:

- If the requested length is not implemented, then the requested length rounded down to the nearest implemented scalable vector length is used.
- Otherwise, the requested length is used.

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
        return ZCR_EL3;
```

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];
```

(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

DBGDTRTText: 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

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

VTCCR: Virtualization Translation Control Register

VTTBR: Virtualization Translation Table Base Register

3020/09/2021 1412:5740

Copyright © 2010-2021 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

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

3020/09/2021 1412:5740

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

APSR, Application Program Status Register

The APSR characteristics are:

Purpose

Hold program status and control information.

Note

Some of the fields in this register are permitted to return the value of the PSTATE field on a read. This is an exception to the general rule that an UNKNOWN field must not return information that cannot be obtained, at the current Privilege level, by an architected mechanism.

For more information see 'The Application Program Status Register, APSR'.

Configuration

This register is present only when AArch32 is supported. Otherwise, direct accesses to APSR are UNDEFINED.

Attributes

APSR 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	
N	Z	C	V	Q	RES0				P	A	N	G	E	R	E	S	RES0				E	A	I	F	RES0				M[4:0]			

N, bit [31]

Negative condition flag. Set to bit[31] of the result of the last flag-setting instruction. If the result is regarded as a two's complement signed integer, then N is set to 1 if the result was negative, and N is set to 0 if the result was positive or zero.

Z, bit [30]

Zero condition flag. Set to 1 if the result of the last flag-setting instruction was zero, and to 0 otherwise. A result of zero often indicates an equal result from a comparison.

C, bit [29]

Carry condition flag. Set to 1 if the last flag-setting instruction resulted in a carry condition, for example an unsigned overflow on an addition.

V, bit [28]

Overflow condition flag. Set to 1 if the last flag-setting instruction resulted in an overflow condition, for example a signed overflow on an addition.

Q, bit [27]

Cumulative saturation bit. Set to 1 to indicate that overflow or saturation occurred in some instructions.

Bits [26:23:20]

Reserved, RES0.

PAN, bit [22]**When FEAT_PAN is implemented:**

Privileged Access Never. This field is UNKNOWN, but is permitted to return the value of PSTATE.PAN field. On writes, this field is treated as Do-Not-Modify.

The reset behavior of this field is:

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

Otherwise:

Reserved, RES0.

Bits [21:20]

Reserved, RES0.

GE, bits [19:16]

Greater than or Equal flags, for parallel addition and subtraction.

Bits [15:10:5]

Reserved, RES0.

E, Bit bit [94]

Endianness. This field is Reserved, UNKNOWNRES1, but is permitted to return the value of PSTATE.E field. On writes, this field is treated as Do-Not-Modify.

The reset behavior of this field is:

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

A, Bits bit [83:0]

SError interrupt mask. This field is UNKNOWN, but is permitted to return the value of PSTATE.A field. On writes, this field is treated as Do-Not-Modify.

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. This field is UNKNOWN, but is permitted to return the value of PSTATE.I field. On writes, this field is treated as Do-Not-Modify.

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. This field is UNKNOWN, but is permitted to return the value of PSTATE.F field. On writes, this field is treated as Do-Not-Modify.

The reset behavior of this field is:

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

Bit [5]

Reserved, RES0.

It is permitted that, on a read of APSR:

- Bit[22] returns the value of PSTATE.PAN
- Bit[9] returns the value of PSTATE.E.
- Bits[8:6] return the value of PSTATE.{A, I, F}, the mask bits.
- Bit[4:0] returns the value of PSTATE.M[4:0]

Note

This is an exception to the general rule that an UNKNOWN field must not return information that cannot be obtained, at the current Privilege level, by an architected mechanism.

For more information see 'The Application Program Status Register, APSR'.

M[4:0], bits [4:0]

Mode. This field is UNKNOWN, but is permitted to return the value of PSTATE.M[4:0] field. On writes, this field is treated as Do-Not-Modify.

The reset behavior of this field is:

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

Accessing APSR

APSR can be read using the MRS instruction and written using the MSR (register) or MSR (immediate) instructions.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ATS12NSOPR, Address Translate Stages 1 and 2 Non-secure Only PL1 Read

The ATS12NSOPR characteristics are:

Purpose

Performs stage 1 and 2 address translations as defined for PL1 and the Non-secure state, with permissions as if reading from the given virtual address.

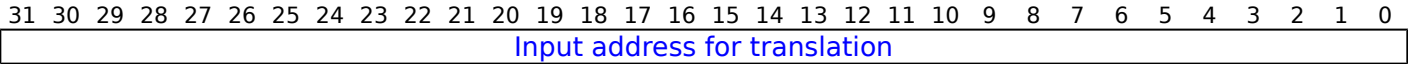
Configuration

This instruction is present only when AArch32 is supported. Otherwise, direct accesses to ATS12NSOPR are UNDEFINED.

Attributes

ATS12NSOPR is a 32-bit System instruction.

Field descriptions



Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This System instruction takes a VA as input. The resulting address is the PA that is the output address of the stage 2 translation.

Executing the ATS12NSOPR 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	0b1000	0b100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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 !ELUsingAArch32(EL2) && SCR_EL3.<NS,EEL2> == '01' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif !ELUsingAArch32(EL3) && SCR_EL3.NS == '0' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch32.ATATS12NSOPR(R[t], TranslationStage_12, EL1, ATAccess_Read);});
elsif PSTATE.EL == EL3 then
    AArch32.ATATS12NSOPR(R[t], TranslationStage_12, EL1, ATAccess_Read);});

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ATS12NSOPW, Address Translate Stages 1 and 2 Non-secure Only PL1 Write

The ATS12NSOPW characteristics are:

Purpose

Performs stage 1 and 2 address translations as defined for PL1 and the Non-secure state, with permissions as if writing to the given virtual address.

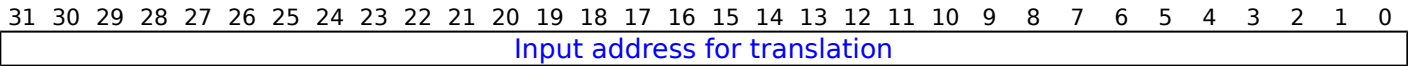
Configuration

This instruction is present only when AArch32 is supported. Otherwise, direct accesses to ATS12NSOPW are UNDEFINED.

Attributes

ATS12NSOPW is a 32-bit System instruction.

Field descriptions



Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This System instruction takes a VA as input. The resulting address is the PA that is the output address of the stage 2 translation.

Executing the ATS12NSOPW 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	0b1000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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 !ELUsingAArch32(EL2) && SCR_EL3.<NS,EEL2> == '01' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif !ELUsingAArch32(EL3) && SCR_EL3.NS == '0' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch32.ATATS12NSOPW(R[t], TranslationStage_12, EL1, ATAccess_Write);});
elsif PSTATE.EL == EL3 then
    AArch32.ATATS12NSOPW(R[t], TranslationStage_12, EL1, ATAccess_Write);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd536e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The ATS12NSOUR characteristics are:

Performs stage 1 and 2 address translations as defined for PL0 and the Non-secure state, with permissions as if reading from the given virtual address.

This instruction is present only when AArch32 is supported. Otherwise, direct accesses to ATS12NSOUR are UNDEFINED.

ATS12NSOUR is a 32-bit System instruction.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Input address for translation																															

Input address for translation. The resulting address can be read from the [PAR](#).

This System instruction takes a VA as input. The resulting address is the PA that is the output address of the stage 2 translation.

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	0b1000	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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 !ELUsingAArch32(EL2) && SCR_EL3.<NS,EEL2> == '01' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif !ELUsingAArch32(EL3) && SCR_EL3.NS == '0' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch32.ATATS12NSOUR(R[t], TranslationStage_12, EL0, ATAccess_Read);});
elsif PSTATE.EL == EL3 then
    AArch32.ATATS12NSOUR(R[t], TranslationStage_12, EL0, ATAccess_Read);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd5b36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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.T7 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T7 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif !ELUsingAArch32(EL2) && SCR_EL3.<NS,EEL2> == '01' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif !ELUsingAArch32(EL3) && SCR_EL3.NS == '0' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    AArch32.ATATS12NSOUW(R[t], TranslationStage_12, EL0, ATAccess_Write);});
elsif PSTATE.EL == EL3 then
    AArch32.ATATS12NSOUW(R[t], TranslationStage_12, EL0, ATAccess_Write);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The ATS1CPR characteristics are:

Purpose

Performs stage 1 address translation as defined for PL1 and the current Security state, with permissions as if reading from the given virtual address.

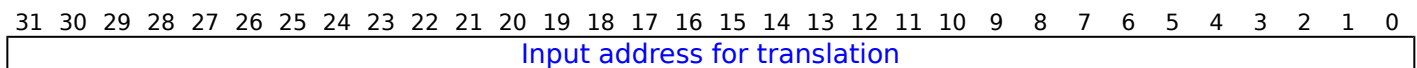
Configuration

This instruction is present only when AArch32 is supported. Otherwise, direct accesses to ATS1CPR are UNDEFINED.

Attributes

ATS1CPR is a 32-bit System instruction.

Field descriptions

**Bits [31:0]**

Input address for translation. The resulting address can be read from the [PAR](#).

This System instruction takes a VA as input. If EL2 is implemented and enabled in the current Security state, the resulting address is the IPA that is the output address of the stage 1 translation. Otherwise, the resulting address is a PA.

Executing the ATS1CPR 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	0b1000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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);
    else
        AArch32.ATATS1CPR(R[t], TranslationStage_1, EL1, ATAccess_Read);});
elsif PSTATE.EL == EL2 then
    AArch32.ATATS1CPR(R[t], TranslationStage_1, EL1, ATAccess_Read);});
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        AArch32.ATATS1CPR(R[t], TranslationStage_1, EL3, ATAccess_Read);
    else
        AArch32.AT(R[t], TranslationStage_1, EL1, ATAccess_Read);});

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ATS1CPRP, Address Translate Stage 1 Current state PL1 Read PAN

The ATS1CPRP characteristics are:

Purpose

Performs a stage 1 address translation at PL1 and in the current Security state, where the value of PSTATE.PAN determines if a read from a location will generate a Permission fault for a privileged access.

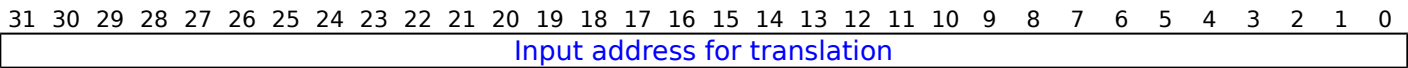
Configuration

This instruction is present only when AArch32 is supported and FEAT_PAN2 is implemented. Otherwise, direct accesses to ATS1CPRP are UNDEFINED.

Attributes

ATS1CPRP is a 32-bit System instruction.

Field descriptions



Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This System instruction takes a VA as input. If EL2 is implemented and enabled in the current Security state, the resulting address is the IPA that is the output address of the stage 1 translation. Otherwise, the resulting address is a PA.

Executing the ATS1CPRP 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	0b1001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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);
    else
        AArch32.ATS1CPRP(R[t], TranslationStage_1, EL1, ATAccess_ReadPAN);});
elsif PSTATE.EL == EL2 then
    AArch32.ATS1CPRP(R[t], TranslationStage_1, EL1, ATAccess_ReadPAN);});
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        AArch32.ATS1CPRP(R[t], TranslationStage_1, EL3, ATAccess_ReadPAN);
    else
        AArch32.AT(R[t], TranslationStage_1, EL1, ATAccess_ReadPAN);});

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ATS1CPW, Address Translate Stage 1 Current state PL1 Write

The ATS1CPW characteristics are:

Purpose

Performs stage 1 address translation as defined for PL1 and the current Security state, with permissions as if writing to the given virtual address.

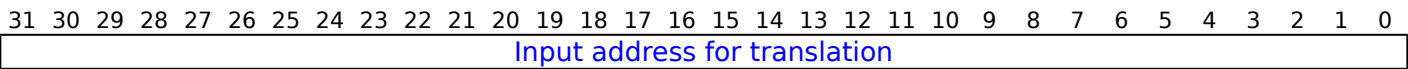
Configuration

This instruction is present only when AArch32 is supported. Otherwise, direct accesses to ATS1CPW are UNDEFINED.

Attributes

ATS1CPW is a 32-bit System instruction.

Field descriptions



Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This System instruction takes a VA as input. If EL2 is implemented and enabled in the current Security state, the resulting address is the IPA that is the output address of the stage 1 translation. Otherwise, the resulting address is a PA.

Executing the ATS1CPW 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	0b1000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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);
    else
        AArch32.ATATS1CPW(R[t], TranslationStage_1, EL1, ATAccess_Write);});
elsif PSTATE.EL == EL2 then
    AArch32.ATATS1CPW(R[t], TranslationStage_1, EL1, ATAccess_Write);});
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        AArch32.ATATS1CPW(R[t], TranslationStage_1, EL3, ATAccess_Write);
    else
        AArch32.AT(R[t], TranslationStage_1, EL1, ATAccess_Write);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ATS1CPWP, Address Translate Stage 1 Current state PL1 Write PAN

The ATS1CPWP characteristics are:

Purpose

Performs a stage 1 address translation at PL1 and in the current Security state, where the value of PSTATE.PAN determines if a write to the location will generate a Permission fault for a privileged access.

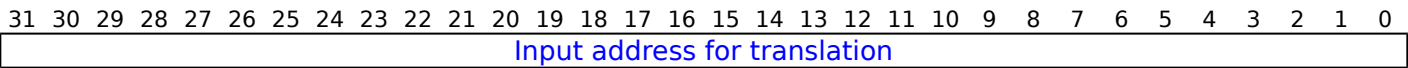
Configuration

This instruction is present only when AArch32 is supported and FEAT_PAN2 is implemented. Otherwise, direct accesses to ATS1CPWP are UNDEFINED.

Attributes

ATS1CPWP is a 32-bit System instruction.

Field descriptions



Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This System instruction takes a VA as input. If EL2 is implemented and enabled in the current Security state, the resulting address is the IPA that is the output address of the stage 1 translation. Otherwise, the resulting address is a PA.

Executing the ATS1CPWP 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	0b1001	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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);
    else
        AArch32.ATATS1CPWP(R[t], TranslationStage_1, EL1, ATAccess_WritePAN);});
elsif PSTATE.EL == EL2 then
    AArch32.ATATS1CPWP(R[t], TranslationStage_1, EL1, ATAccess_WritePAN);});
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        AArch32.ATATS1CPWP(R[t], TranslationStage_1, EL3, ATAccess_WritePAN);
    else
        AArch32.AT(R[t], TranslationStage_1, EL1, ATAccess_WritePAN);});

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ATS1CUR, Address Translate Stage 1 Current state Unprivileged Read

The ATS1CUR characteristics are:

Purpose

Performs stage 1 address translation as defined for PL0 and the current Security state, with permissions as if reading from the given virtual address.

Configuration

This instruction is present only when AArch32 is supported. Otherwise, direct accesses to ATS1CUR are UNDEFINED.

Attributes

ATS1CUR 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
Input address for translation																															

Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This System instruction takes a VA as input. If EL2 is implemented and enabled in the current Security state, the resulting address is the IPA that is the output address of the stage 1 translation. Otherwise, the resulting address is a PA.

Executing the ATS1CUR 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	0b1000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T7 == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        AArch32.ATS1CUR(R[t], TranslationStage_1, EL0, ATAccess_Read);
elseif PSTATE.EL == EL2 then
    AArch32.ATS1CUR(R[t], TranslationStage_1, EL0, ATAccess_Read);
elseif PSTATE.EL == EL3 then
    AArch32.ATS1CUR(R[t], TranslationStage_1, EL0, ATAccess_Read);

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ATS1CUW, Address Translate Stage 1 Current state Unprivileged Write

The ATS1CUW characteristics are:

Purpose

Performs stage 1 address translation as defined for PL0 and the current Security state, with permissions as if writing to the given virtual address.

Configuration

This instruction is present only when AArch32 is supported. Otherwise, direct accesses to ATS1CUW are UNDEFINED.

Attributes

ATS1CUW 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
Input address for translation																															

Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This System instruction takes a VA as input. If EL2 is implemented and enabled in the current Security state, the resulting address is the IPA that is the output address of the stage 1 translation. Otherwise, the resulting address is a PA.

Executing the ATS1CUW 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	0b1000	0b011

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T7 == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        AArch32.ATS1CUW(R[t], TranslationStage_1, EL0, ATAccess_Write);
elseif PSTATE.EL == EL2 then
    AArch32.ATS1CUW(R[t], TranslationStage_1, EL0, ATAccess_Write);
elseif PSTATE.EL == EL3 then
    AArch32.ATS1CUW(R[t], TranslationStage_1, EL0, ATAccess_Write);

```

3020/09/2021 14:12:52.37: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

(old)

htmldiff from-

(new)

ATS1HR, Address Translate Stage 1 Hyp mode Read

The ATS1HR characteristics are:

Purpose

Performs stage 1 address translation as defined for PL2 and the Non-secure state, with permissions as if reading from the given virtual address.

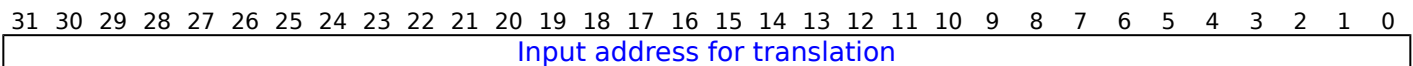
Configuration

This instruction is present only when AArch32 is supported. Otherwise, direct accesses to ATS1HR are UNDEFINED.

Attributes

ATS1HR is a 32-bit System instruction.

Field descriptions



Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This System instruction takes a VA as input. The resulting address is the PA that is the output address of the translation.

Executing the ATS1HR instruction

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

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	0b100	0b0111	0b1000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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);
    else
        UNDEFINED;ATS1HR(R[t]);
elsif PSTATE.EL == EL2 then
    AArch32.ATATS1HR(R[t], TranslationStage_1, EL2, ATAccess_Read);});
elsif PSTATE.EL == EL3 then
    AArch32.ATATS1HR(R[t], TranslationStage_1, EL2, ATAccess_Read);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

ATS1HW, Address Translate Stage 1 Hyp mode Write

The ATS1HW characteristics are:

Purpose

Performs stage 1 address translation as defined for PL2 and the Non-secure state, with permissions as if writing to the given virtual address.

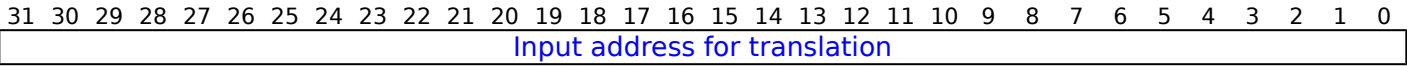
Configuration

This instruction is present only when AArch32 is supported. Otherwise, direct accesses to ATS1HW are UNDEFINED.

Attributes

ATS1HW is a 32-bit System instruction.

Field descriptions



Bits [31:0]

- Input address for translation. The resulting address can be read from the [PAR](#).
- This System instruction takes a VA as input. The resulting address is the PA that is the output address of the translation.

Executing the ATS1HW instruction

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

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	0b100	0b0111	0b1000	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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);
    else
        UNDEFINED;ATS1HW(R[t]);
elsif PSTATE.EL == EL2 then
    AArch32.ATATS1HW(R[t], TranslationStage_1, EL2, ATAccess_Write);});
elsif PSTATE.EL == EL3 then
    AArch32.ATATS1HW(R[t], TranslationStage_1, EL2, ATAccess_Write);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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				GVMID		NS	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, 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 ASID 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);
    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' && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HFGITR_EL2.CFPRCTX == '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.RestrictPredictionCFPRCTX(R[t], RestrictType_ControlFlow);});
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.RestrictPredictionCFPRCTX(R[t], RestrictType_ControlFlow);});
elseif PSTATE.EL == EL2 then
    AArch32.RestrictPredictionCFPRCTX(R[t], RestrictType_ControlFlow);});
elseif PSTATE.EL == EL3 then
    AArch32.RestrictPredictionCFPRCTX(R[t], RestrictType_ControlFlow);});

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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 AArch32 is supported. 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.

The possible values are:

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.

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.

LoUIS, bits [23:21]

Level of Unification Inner Shareable for the cache hierarchy.

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

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;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T0 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TID2 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TID4 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TID2 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HCR2.TID4 == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        return CLIDR;
elseif PSTATE.EL == EL2 then
    return CLIDR;
elseif PSTATE.EL == EL3 then
    return CLIDR;

```


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

(old)

htmldiff from-

(new)

CNTHCTL, Counter-timer Hyp Control register

The CNTHCTL characteristics are:

Purpose

Controls the generation of an event stream from the physical counter, and access from Non-secure EL1 modes to the physical counter and the Non-secure EL1 physical timer.

Configuration

AArch32 System register CNTHCTL bits [31:0] are architecturally mapped to AArch64 System register [CNTHCTL_EL2\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to CNTHCTL are UNDEFINED.

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

Attributes

CNTHCTL 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														EVNTIS	RES0														EVNTI	EVNTDIR	EVNTEN	PL1PCEN	PL1PCTEN

Bits [31:18]

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.EVNTI field applies to CNTPCT[15:0] .
0b1	The CNTHCTL.EVNTI field applies to CNTPCT[23:8] .

The reset behavior of this field is:

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

Otherwise:

Reserved, RES0.

Bits [16:8]

Reserved, RES0.

EVNTI, bits [7:4]

Selects which bit of ~~the counter register~~ **CNTPCT**, 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.EVNTIS is 1, this field selects a trigger bit in the range 8 to 23 of ~~the counter register~~ **CNTPCT**. ~~is the trigger.~~

Otherwise, this field selects a trigger bit in the range 0 to 15 of ~~the counter register~~ **CNTPCT**.

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 ~~counter register~~ **CNTPCT** trigger bit, as seen from EL2 and defined by EVNTI, generates an event when the event stream is ~~enabled~~ **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 ~~the counter register~~ **CNTPCT** 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.

PL1PCEN, bit [1]

Traps Non-secure EL0 and EL1 accesses to the physical timer registers to Hyp mode.

PL1PCEN	Meaning
0b0	Non-secure EL0 and EL1 accesses to the CNTP_CTL , CNTP_CVAL , and CNTP_TVAL are trapped to Hyp mode, unless the it is trapped by CNTKCTL .PLOPTEN.
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.

PL1PCTEN, bit [0]

Traps Non-secure EL0 and EL1 accesses to the physical counter register to Hyp mode.

PL1PCTEN	Meaning
0b0	Non-secure EL0 and EL1 accesses to the CNTPCT are trapped to Hyp mode, unless it is trapped by 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

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	0b1110	0b0001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    return CNTHCTL;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        return CNTHCTL;
```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b100	0b1110	0b0001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    CNTHCTL = R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        CNTHCTL = R[t];
```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTHP_CTL, Counter-timer Hyp Physical Timer Control register

The CNTHP_CTL characteristics are:

Purpose

Control register for the Hyp mode physical timer.

Configuration

AArch32 System register CNTHP_CTL bits [31:0] are architecturally mapped to AArch64 System register [CNTHP_CTL_EL2\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to CNTHP_CTL are UNDEFINED.

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

Attributes

CNTHP_CTL 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																													ISTATUS	IMASK	ENABLE

Bits [31:3]

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 [CNTHP_TVAL](#) 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, when the PE resets into EL2 or EL3, this field resets to 0.

Accessing CNTHP_CTL

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	0b1110	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    return CNTHP_CTL;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        return CNTHP_CTL;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b100	0b1110	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    CNTHP_CTL = R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        CNTHP_CTL = R[t];

```

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b0010	0b001

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
        elsif ELUsingAArch32(EL1) && CNTKCTL.PL0PTEN == '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 == '0' && CNTHCTL_EL2.EL1PCEN == '0'
    then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN == '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        return CNTHPS_CTL_EL2;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
    then
        return CNTHP_CTL_EL2;
    else
        return CNTP_CTL;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0'
    then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0'
    then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x03);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
        return CNTP_CTL_NS;
    else
        return CNTP_CTL;
    elsif PSTATE.EL == EL2 then
        if HaveEL(EL3) && ELUsingAArch32(EL3) then
            return CNTP_CTL_NS;
        else
            return CNTP_CTL;
    elsif PSTATE.EL == EL3 then
        if SCR.NS == '0' then
            return CNTP_CTL_S;
        else
            return CNTP_CTL_NS;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b0010	0b001


```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0PTEN == '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 == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
        CNTHPS_CTL_EL2 = R[t];
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
        CNTHP_CTL_EL2 = R[t];
    else
        CNTP_CTL = R[t];
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x03);
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) then
        CNTP_CTL_NS = R[t];
    else
        CNTP_CTL = R[t];
elseif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        CNTP_CTL_NS = R[t];
    else
        CNTP_CTL = R[t];
elseif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        CNTP_CTL_S = R[t];
    else
        CNTP_CTL_NS = R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

CNTHP_CVAL, Counter-timer Hyp Physical CompareValue register

The CNTHP_CVAL characteristics are:

Purpose

Holds the compare value for the Hyp mode physical timer.

Configuration

AArch32 System register CNTHP_CVAL bits [63:0] are architecturally mapped to AArch64 System register [CNTHP_CVAL_EL2\[63:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to CNTHP_CVAL are UNDEFINED.

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

Attributes

CNTHP_CVAL is 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 EL2 physical timer CompareValue.

When [CNTHP_CTL](#).ENABLE is 1, the timer condition is met when ([CNTPCT](#) - 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:

- [CNTHP_CTL](#).ISTATUS is set to 1.
- If [CNTHP_CTL](#).IMASK is 0, an interrupt is generated.

When [CNTHP_CTL](#).ENABLE is 0, the timer condition is not met, but [CNTPCT](#) 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 CNTHP_CVAL

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

MRRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm>

coproc	CRm	opc1
0b1111	0b1110	0b0110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    return CNTHP_CVAL;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        return CNTHP_CVAL;

```

MCRR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm>

coproc	CRm	opc1
0b1111	0b1110	0b0110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    CNTHP_CVAL = R[t2]:R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        CNTHP_CVAL = R[t2]:R[t];

```

MRRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm>

coproc	CRm	opc1
0b1111	0b1110	0b0010

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x04);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x04);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0PTEN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x04);
        elseif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN == '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x04);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0' && IsFeatureImplemented(FEAT_SEL2) then
        return CNTHPS_CVAL_EL2;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1' then
        return CNTHP_CVAL_EL2;
    else
        return CNTP_CVAL;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x04);
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) then
        return CNTP_CVAL_NS;
    else
        return CNTP_CVAL;
elseif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        return CNTP_CVAL_NS;
    else
        return CNTP_CVAL;
elseif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        return CNTP_CVAL_S;
    else
        return CNTP_CVAL_NS;

```

MCCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm>

coproc	CRm	opc1
0b1111	0b1110	0b0010

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x04);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x04);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0PTEN == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x04);
        elseif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TGE == '1' then
            AArch32.TakeHypTrapException(0x00);
        else
            UNDEFINED;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x04);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
        CNTHPS_CVAL_EL2 = R[t2]:R[t];
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
        CNTHP_CVAL_EL2 = R[t2]:R[t];
    else
        CNTP_CVAL = R[t2]:R[t];
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x04);
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) then
        CNTP_CVAL_NS = R[t2]:R[t];
    else
        CNTP_CVAL = R[t2]:R[t];
elseif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        CNTP_CVAL_NS = R[t2]:R[t];
    else
        CNTP_CVAL = R[t2]:R[t];
elseif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        CNTP_CVAL_S = R[t2]:R[t];
    else
        CNTP_CVAL_NS = R[t2]:R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTHP_TVAL, Counter-timer Hyp Physical Timer TimerValue register

The CNTHP_TVAL characteristics are:

Purpose

Holds the timer value for the Hyp mode physical timer.

Configuration

AArch32 System register CNTHP_TVAL bits [31:0] are architecturally mapped to AArch64 System register [CNTHP_TVAL_EL2\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to CNTHP_TVAL are UNDEFINED.

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

Attributes

CNTHP_TVAL 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
TimerValue																															

TimerValue, bits [31:0]

The TimerValue view of the EL2 physical timer.

On a read of this register:

- If [CNTHP_CTL.ENABLE](#) is 0, the value returned is UNKNOWN.
- If [CNTHP_CTL.ENABLE](#) is 1, the value returned is ([CNTHP_CVAL](#) - [CNTPCT](#)).

On a write of this register, [CNTHP_CVAL](#) is set to ([CNTPCT](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTHP_CTL.ENABLE](#) is 1, the timer condition is met when ([CNTPCT](#) - [CNTHP_CVAL](#)) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTHP_CTL.ISTATUS](#) is set to 1.
- If [CNTHP_CTL.IMASK](#) is 0, an interrupt is generated.

When [CNTHP_CTL.ENABLE](#) is 0, the timer condition is not met, but [CNTPCT](#) 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 CNTHP_TVAL

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	0b1110	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if CNTHP_CTL.ENABLE == '0' then
        return bits(32) UNKNOWNCNTHP_TVAL;
    else
        return (CNTHP_CVAL - PhysicalCountInt())<31:0>;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        if CNTHP_CTL.ENABLE == '0' then
            return bits(32) UNKNOWNCNTHP_TVAL;
        else
            return (CNTHP_CVAL - PhysicalCountInt())<31:0>;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b100	0b1110	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    CNTHP_CVALCNTHP_TVAL = SignExtend(R[t],64) + PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    if CNTHP_TVAL SCR.NS == '0' then
        UNDEFINED;
    else
        CNTHP_CVAL = SignExtend(R[t],64) + PhysicalCountInt();

```

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b0010	0b000

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0PTEN == '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 == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            return bits(32) UNKNOWN;CNTHPS_TVAL_EL2;
        else
            return (CNTHPS_CVAL_EL2 - PhysicalCountInt())<31:0>;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
        if CNTHP_CTL_EL2.ENABLE == '0' then
            return bits(32) UNKNOWN;CNTHP_TVAL_EL2;
        else
            return (CNTHP_CVAL_EL2 - PhysicalCountInt())<31:0>;
    elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && !ELUsingAArch32(EL2) && SCR_EL3.ECVEn
== '1' && CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11' then
        if CNTP_CTL.ENABLE == '0' then
            return bits(32) UNKNOWN;
        else
            return (CNTP_CVAL - (PhysicalCountInt() - CNTPOFF_EL2))<31:0>;
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) then
        if SCR.NS == '1' then
            if CNTP_CTL_NS.ENABLE == '0' then
                return bits(32) UNKNOWN;CNTP_TVAL;
            else
                return (CNTP_CVAL_NS - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL_S.ENABLE == '0' then
                return bits(32) UNKNOWN;
            else
                return (CNTP_CVAL_S - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL.ENABLE == '0' then
                return bits(32) UNKNOWN;
            else
                return (CNTP_CVAL - PhysicalCountInt())<31:0>;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0'
then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
            AArch32.TakeHypTrapException(0x03);
        elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && !ELUsingAArch32(EL2) && SCR_EL3.ECVEn
== '1' && CNTHCTL_EL2.ECV == '1' then

```



```

    if CNTP_CTL.ENABLE == '0' then
        return bits(32) UNKNOWN;
    else
        return (CNTP_CVAL - (PhysicalCountInt() - CNTPOFF_EL2))<31:0>;
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
        if CNTP_CTL_NS.ENABLE == '0' then
            return bits(32) UNKNOWNCNTP_TVAL_NS;
        else
            return (CNTP_CVAL_NS - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL;
            else
                return (CNTP_CVAL - PhysicalCountInt())<31:0>;
    elsif PSTATE.EL == EL2 then
        if HaveEL(EL3) && ELUsingAArch32(EL3) then
            if CNTP_CTL_NS.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL_NS;
            else
                return (CNTP_CVAL_NS - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL;
            else
                return (CNTP_CVAL - PhysicalCountInt())<31:0>;
    elsif PSTATE.EL == EL3 then
        if SCR.NS == '0' then
            if CNTP_CTL_S.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL_S;
            else
                return (CNTP_CVAL_S - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL_NS.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL_NS;
            else
                return (CNTP_CVAL_NS - PhysicalCountInt())<31:0>;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b0010	0b000

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0PTEN == '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 == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
        CNTHPS_CVAL_EL2CNTHPS_TVAL_EL2 = SignExtend(R[t],64) + PhysicalCountInt();};
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
        CNTHP_CVAL_EL2CNTHP_TVAL_EL2 = SignExtend(R[t],64) + PhysicalCountInt();};
        elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && !ELUsingAArch32(EL2) && SCR_EL3.ECVEn
== '1' && CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11' then
            CNTP_CVAL = SignExtend(R[t],64) + PhysicalCountInt() - CNTPOFF_EL2;
        elseif HaveEL(EL3) && ELUsingAArch32(EL3) then
            if SCR.NS == '1' then
                CNTP_CVAL_NS = SignExtend(R[t],64) + PhysicalCountInt();
            else
                CNTP_CVAL_S = SignExtend(R[t],64) + PhysicalCountInt();
            else
                CNTP_CVALCNTP_TVAL = SignExtend(R[t],64) + PhysicalCountInt();};
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x03);
    elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && !ELUsingAArch32(EL2) && SCR_EL3.ECVEn
== '1' && CNTHCTL_EL2.ECV == '1' then
        CNTP_CVAL = SignExtend(R[t],64) + PhysicalCountInt() - CNTPOFF_EL2;
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) then
        CNTP_CVAL_NSCNTP_TVAL_NS = SignExtend(R[t],64) + PhysicalCountInt();};
    else
        CNTP_CVALCNTP_TVAL = SignExtend(R[t],64) + PhysicalCountInt();};
elseif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        CNTP_CVAL_NSCNTP_TVAL_NS = SignExtend(R[t],64) + PhysicalCountInt();};
    else
        CNTP_CVALCNTP_TVAL = SignExtend(R[t],64) + PhysicalCountInt();};
elseif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        CNTP_CVAL_SCNTP_TVAL_S = SignExtend(R[t],64) + PhysicalCountInt();};
    else
        CNTP_CVAL_NSCNTP_TVAL_NS = SignExtend(R[t],64) + PhysicalCountInt();};

```

3020/09/2021 14:12:52: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTHPS_TVAL, Counter-timer Secure Physical Timer TimerValue Register (EL2)

The CNTHPS_TVAL characteristics are:

Purpose

Provides AArch32 access from EL0 to the timer value for the Secure EL2 physical timer.

Configuration

AArch32 System register CNTHPS_TVAL bits [31:0] are architecturally mapped to AArch64 System register [CNTHPS_TVAL_EL2\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_SEL2 is implemented. Otherwise, direct accesses to CNTHPS_TVAL are UNDEFINED.

Attributes

CNTHPS_TVAL 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
																TimerValue															

TimerValue, bits [31:0]

The TimerValue view of the EL2 physical timer.

On a read of this register:

- If [CNTHPS_CTL_EL2.ENABLE](#) is 0, the value returned is UNKNOWN.
- If [CNTHPS_CTL_EL2.ENABLE](#) is 1, the value returned is ([CNTHPS_CVAL_EL2](#) - [CNTPCT_EL0](#)).

On a write of this register, [CNTHPS_CVAL_EL2](#) is set to ([CNTPCT_EL0](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTHPS_CTL_EL2.ENABLE](#) is 1, the timer condition is met when ([CNTPCT_EL0](#) - [CNTHPS_CVAL_EL2](#)) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTHPS_CTL_EL2.ISTATUS](#) is set to 1.
- If [CNTHPS_CTL_EL2.IMASK](#) is 0, an interrupt is generated.

When [CNTHPS_CTL_EL2.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 CNTHPS_TVAL

This register is accessed using the encoding for [CNTP_TVAL](#).

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	0b0010	0b000

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN
    == '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
        elsif ELUsingAArch32(EL1) && CNTKCTL.PL0PTEN == '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 == '0' && CNTHCTL_EL2.EL1PCEN == '0'
    then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN
    == '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN
    == '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
    && IsFeatureImplemented(FEAT_SEL2) then
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            return bits(32) UNKNOWN;CNTHPS_TVAL_EL2;
        else
            return (CNTHPS_CVAL_EL2 - PhysicalCountInt())<31:0>;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
    then
        if CNTHP_CTL_EL2.ENABLE == '0' then
            return bits(32) UNKNOWN;CNTHP_TVAL_EL2;
        else
            return (CNTHP_CVAL_EL2 - PhysicalCountInt())<31:0>;
        elsif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && !ELUsingAArch32(EL2) && SCR_EL3.ECVEn
    == '1' && CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11' then
        if CNTP_CTL.ENABLE == '0' then
            return bits(32) UNKNOWN;
        else
            return (CNTP_CVAL - (PhysicalCountInt() - CNTPOFF_EL2))<31:0>;
        elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
            if SCR.NS == '1' then
                if CNTP_CTL_NS.ENABLE == '0' then
                    return bits(32) UNKNOWN;CNTP_TVAL;
                else
                    return (CNTP_CVAL_NS - PhysicalCountInt())<31:0>;
            else
                if CNTP_CTL_S.ENABLE == '0' then
                    return bits(32) UNKNOWN;
                else
                    return (CNTP_CVAL_S - PhysicalCountInt())<31:0>;
            else
                if CNTP_CTL.ENABLE == '0' then
                    return bits(32) UNKNOWN;
                else
                    return (CNTP_CVAL - PhysicalCountInt())<31:0>;
        elsif PSTATE.EL == EL1 then
            if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0'
    then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0'
    then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
                AArch32.TakeHypTrapException(0x03);
            elsif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && !ELUsingAArch32(EL2) && SCR_EL3.ECVEn
    == '1' && CNTHCTL_EL2.ECV == '1' then

```

```

    if CNTP_CTL.ENABLE == '0' then
        return bits(32) UNKNOWN;
    else
        return (CNTP_CVAL - (PhysicalCountInt() - CNTPOFF_EL2))<31:0>;
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
        if CNTP_CTL_NS.ENABLE == '0' then
            return bits(32) UNKNOWNCNTP_TVAL_NS;
        else
            return (CNTP_CVAL_NS - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL;
            else
                return (CNTP_CVAL - PhysicalCountInt())<31:0>;
    elsif PSTATE.EL == EL2 then
        if HaveEL(EL3) && ELUsingAArch32(EL3) then
            if CNTP_CTL_NS.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL_NS;
            else
                return (CNTP_CVAL_NS - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL;
            else
                return (CNTP_CVAL - PhysicalCountInt())<31:0>;
    elsif PSTATE.EL == EL3 then
        if SCR.NS == '0' then
            if CNTP_CTL_S.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL_S;
            else
                return (CNTP_CVAL_S - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL_NS.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL_NS;
            else
                return (CNTP_CVAL_NS - PhysicalCountInt())<31:0>;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b0010	0b000

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
        elsif ELUsingAArch32(EL1) && CNTKCTL.PL0PTEN == '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 == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN
== '0' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN
== '0' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elsif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
            AArch32.TakeHypTrapException(0x03);
        elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
            CNTHPS_CVAL_EL2CNTHPS_TVAL_EL2 = SignExtend(R[t],64) + PhysicalCountInt();};
        elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
            CNTHP_CVAL_EL2CNTHP_TVAL_EL2 = SignExtend(R[t],64) + PhysicalCountInt();};
            elsif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && !ELUsingAArch32(EL2) && SCR_EL3.ECVEn
== '1' && CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11' then
                CNTP_CVAL = SignExtend(R[t],64) + PhysicalCountInt() - CNTPOFF_EL2;
            elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
                if SCR.NS == '1' then
                    CNTP_CVAL_NS = SignExtend(R[t],64) + PhysicalCountInt();
                else
                    CNTP_CVAL_S = SignExtend(R[t],64) + PhysicalCountInt();
                else
                    CNTP_CVALCNTP_TVAL = SignExtend(R[t],64) + PhysicalCountInt();};
        elsif PSTATE.EL == EL1 then
            if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0'
then
                AArch64.AArch32SystemAccessTrap(EL2, 0x03);
            elsif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
                AArch32.TakeHypTrapException(0x03);
            elsif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && !ELUsingAArch32(EL2) && SCR_EL3.ECVEn
== '1' && CNTHCTL_EL2.ECV == '1' then
                CNTP_CVAL = SignExtend(R[t],64) + PhysicalCountInt() - CNTPOFF_EL2;
            elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
                CNTP_CVAL_NSCNTP_TVAL_NS = SignExtend(R[t],64) + PhysicalCountInt();};
            else
                CNTP_CVALCNTP_TVAL = SignExtend(R[t],64) + PhysicalCountInt();};
        elsif PSTATE.EL == EL2 then
            if HaveEL(EL3) && ELUsingAArch32(EL3) then
                CNTP_CVAL_NSCNTP_TVAL_NS = SignExtend(R[t],64) + PhysicalCountInt();};
            else
                CNTP_CVALCNTP_TVAL = SignExtend(R[t],64) + PhysicalCountInt();};
        elsif PSTATE.EL == EL3 then
            if SCR.NS == '0' then
                CNTP_CVAL_SCNTP_TVAL_S = SignExtend(R[t],64) + PhysicalCountInt();};
            else
                CNTP_CVAL_NSCNTP_TVAL_NS = SignExtend(R[t],64) + PhysicalCountInt();};

```


3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

(old)

htmldiff from-

(new)

CNTHV_TVAL, Counter-timer Virtual Timer TimerValue register (EL2)

The CNTHV_TVAL characteristics are:

Purpose

Provides AArch32 access to the timer value for the EL2 virtual timer.

Configuration

AArch32 System register CNTHV_TVAL bits [31:0] are architecturally mapped to AArch64 System register [CNTHV_TVAL_EL2\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_VHE is implemented. Otherwise, direct accesses to CNTHV_TVAL are UNDEFINED.

Attributes

CNTHV_TVAL 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
																TimerValue															

TimerValue, bits [31:0]

The TimerValue view of the EL2 virtual timer.

On a read of this register:

- If [CNTHV_CTL.ENABLE](#) is 0, the value returned is UNKNOWN.
- If [CNTHV_CTL.ENABLE](#) is 1, the value returned is ([CNTHV_CVAL](#) - [CNTVCT](#)).

On a write of this register, [CNTHV_CVAL](#) is set to ([CNTVCT](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTHV_CTL.ENABLE](#) is 1, the timer condition is met when ([CNTVCT](#) - [CNTHV_CVAL](#)) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTHV_CTL.ISTATUS](#) is set to 1.
- If [CNTHV_CTL.IMASK](#) is 0, an interrupt is generated.

When [CNTHV_CTL.ENABLE](#) is 0, the timer condition is not met, but [CNTVCT](#) continues to count, so the TimerValue view appears to continue to count down.

Accessing CNTHV_TVAL

This register is accessed using the encoding for [CNTV_TVAL](#).

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	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
        if CNTHVS_CTL_EL2.ENABLE == '0' then
            return bits(32) UNKNOWN<CNTHVS_TVAL_EL2>;
        else
            return (CNTHVS_CVAL_EL2 - PhysicalCountInt())<31:0>;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
        if CNTHV_CTL_EL2.ENABLE == '0' then
            return bits(32) UNKNOWN<CNTHV_TVAL_EL2>;
        else
            return (CNTHV_CVAL_EL2 - PhysicalCountInt())<31:0>;
    else
        if CNTV_CTL.ENABLE == '0' then
            return bits(32) UNKNOWN<CNTV_TVAL>;
        elseif HaveEL(EL2) && !ELUsingAArch32(EL2) then
            return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF_EL2))<31:0>;
        elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
            return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF))<31:0>;
        else
            return (CNTV_CVAL - PhysicalCountInt())<31:0>;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && CNTHCTL_EL2.EL1TVT == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            if CNTV_CTL.ENABLE == '0' then
                return bits(32) UNKNOWN<CNTV_TVAL>;
            elseif HaveEL(EL2) && !ELUsingAArch32(EL2) then
                return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF_EL2))<31:0>;
            elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
                return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF))<31:0>;
            else
                return (CNTV_CVAL - PhysicalCountInt())<31:0>;
    elseif PSTATE.EL == EL2 then
        if CNTV_CTL.ENABLE == '0' then
            return bits(32) UNKNOWN<CNTV_TVAL>;
        else
            return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF))<31:0>;
    elseif PSTATE.EL == EL3 then
        if CNTV_CTL.ENABLE == '0' then
            return bits(32) UNKNOWN<CNTV_TVAL>;
        elseif HaveEL(EL2) then
            return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF))<31:0>;
        else
            return (CNTV_CVAL - PhysicalCountInt())<31:0>;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2CNTHVS_TVAL_EL2 = SignExtend(R[t],64) + PhysicalCountInt();
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
        CNTHV_CVAL_EL2CNTHV_TVAL_EL2 = SignExtend(R[t],64) + PhysicalCountInt();
    else
        ifCNTV_TVAL HaveEL(EL2) && !ELUsingAArch32(EL2) then
            CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt() - CNTVOFF_EL2;
        elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
            CNTV_CVAL = SignExtend(R[t], 64) + PhysicalCountInt() - CNTVOFF;
        else
            CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt();
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && CNTHCTL_EL2.EL1TVT == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            ifCNTV_TVAL HaveEL(EL2) && !ELUsingAArch32(EL2) then
                CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt() - CNTVOFF_EL2;
            elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
                CNTV_CVAL = SignExtend(R[t], 64) + PhysicalCountInt() - CNTVOFF;
            else
                CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt();
    elseif PSTATE.EL == EL2 then
        CNTV_CVALCNTV_TVAL = SignExtend(R[t], 64) + PhysicalCountInt() - CNTVOFF;
    elseif PSTATE.EL == EL3 then
        ifCNTV_TVAL HaveEL(EL2) then
            CNTV_CVAL = SignExtend(R[t], 64) + PhysicalCountInt() - CNTVOFF;
        else
            CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt();

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTHVS_TVAL, Counter-timer Secure Virtual Timer TimerValue Register (EL2)

The CNTHVS_TVAL characteristics are:

Purpose

Provides AArch32 access to the timer value for the Secure EL2 virtual timer.

Configuration

AArch32 System register CNTHVS_TVAL bits [31:0] are architecturally mapped to AArch64 System register [CNTHVS_TVAL_EL2\[31:0\]](#).

This register is present only when AArch32 is supported and FEAT_SEL2 is implemented. Otherwise, direct accesses to CNTHVS_TVAL are UNDEFINED.

Attributes

CNTHVS_TVAL 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
																TimerValue															

TimerValue, bits [31:0]

The TimerValue view of the EL2 virtual timer.

On a read of this register:

- If [CNTHVS_CTL.ENABLE](#) is 0, the value returned is UNKNOWN.
- If [CNTHVS_CTL.ENABLE](#) is 1, the value returned is ([CNTHVS_CVAL](#) - [CNTVCT](#)).

On a write of this register, [CNTHVS_CVAL](#) is set to ([CNTVCT](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTHVS_CTL.ENABLE](#) is 1, the timer condition is met when ([CNTVCT](#) - [CNTHVS_CVAL](#)) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTHVS_CTL.ISTATUS](#) is set to 1.
- If [CNTHVS_CTL.IMASK](#) is 0, an interrupt is generated.

When [CNTHVS_CTL.ENABLE](#) is 0, the timer condition is not met, but [CNTVCT](#) continues to count, so the TimerValue view appears to continue to count down.

Accessing CNTHVS_TVAL

This register is accessed using the encoding for [CNTV_TVAL](#).

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	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
        if CNTHVS_CTL_EL2.ENABLE == '0' then
            return bits(32) UNKNOWNCNTHVS_TVAL_EL2;
        else
            return (CNTHVS_CVAL_EL2 - PhysicalCountInt())<31:0>;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
        if CNTHV_CTL_EL2.ENABLE == '0' then
            return bits(32) UNKNOWNCNTHV_TVAL_EL2;
        else
            return (CNTHV_CVAL_EL2 - PhysicalCountInt())<31:0>;
    else
        if CNTV_CTL.ENABLE == '0' then
            return bits(32) UNKNOWNCNTV_TVAL;
        elseif HaveEL(EL2) && !ELUsingAArch32(EL2) then
            return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF_EL2))<31:0>;
        elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
            return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF))<31:0>;
        else
            return (CNTV_CVAL - PhysicalCountInt())<31:0>;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && CNTHCTL_EL2.EL1TVT == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            if CNTV_CTL.ENABLE == '0' then
                return bits(32) UNKNOWNCNTV_TVAL;
            elseif HaveEL(EL2) && !ELUsingAArch32(EL2) then
                return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF_EL2))<31:0>;
            elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
                return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF))<31:0>;
            else
                return (CNTV_CVAL - PhysicalCountInt())<31:0>;
    elseif PSTATE.EL == EL2 then
        if CNTV_CTL.ENABLE == '0' then
            return bits(32) UNKNOWNCNTV_TVAL;
        else
            return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF))<31:0>;
    elseif PSTATE.EL == EL3 then
        if CNTV_CTL.ENABLE == '0' then
            return bits(32) UNKNOWNCNTV_TVAL;
        elseif HaveEL(EL2) then
            return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF))<31:0>;
        else
            return (CNTV_CVAL - PhysicalCountInt())<31:0>;

```


MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2CNTHVS_TVAL_EL2 = SignExtend(R[t],64) + PhysicalCountInt();;;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
        CNTHV_CVAL_EL2CNTHV_TVAL_EL2 = SignExtend(R[t],64) + PhysicalCountInt();;;
    else
        ifCNTV_TVAL HaveEL(EL2) && !ELUsingAArch32(EL2) then
            CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt() - CNTVOFF_EL2 ;
        elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
            CNTV_CVAL = SignExtend(R[t], 64) + PhysicalCountInt() - CNTVOFF;
        else
            CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt();;;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && CNTHCTL_EL2.EL1TVT == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            ifCNTV_TVAL HaveEL(EL2) && !ELUsingAArch32(EL2) then
                CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt() - CNTVOFF_EL2 ;
            elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
                CNTV_CVAL = SignExtend(R[t], 64) + PhysicalCountInt() - CNTVOFF;
            else
                CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt();;;
    elseif PSTATE.EL == EL2 then
        CNTV_CVALCNTV_TVAL = SignExtend(R[t], 64) + PhysicalCountInt() - CNTVOFF;;;
    elseif PSTATE.EL == EL3 then
        ifCNTV_TVAL HaveEL(EL2) then
            CNTV_CVAL = SignExtend(R[t], 64) + PhysicalCountInt() - CNTVOFF;
        else
            CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt();;;

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTKCTL, Counter-timer Kernel Control register

The CNTKCTL characteristics are:

Purpose

Controls the generation of an event stream from the virtual counter, and access from EL0 modes to the physical counter, virtual counter, EL1 physical timers, and the virtual timer.

Configuration

AArch32 System register CNTKCTL bits [31:0] are architecturally mapped to AArch64 System register [CNTKCTL_EL1\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to CNTKCTL are UNDEFINED.

Attributes

CNTKCTL 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														EVNTIS	RES0				PLOPTEN	PLOVTEN	EVNTI	EVNTDIR	EVNTEN	PLOVCTEN	PLOPCTEN						

Bits [31:18]

Reserved, RES0.

EVENTIS, bit [17]

When FEAT_ECV is implemented:

Controls the scale of the generation of the event stream.

EVENTIS	Meaning
0b0	The CNTKCTL.EVNTI field applies to CNTVCT[15:0] .
0b1	The CNTKCTL.EVNTI field applies to CNTVCT[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.

Bits [16:10]

Reserved, RES0.

PLOPTEN, bit [9]

Traps PL0 accesses to the physical timer registers to Undefined mode.

PLOPTEN	Meaning
0b0	PL0 accesses to the CNTP_CTL , CNTP_CVAL , and CNTP_TVAL registers are trapped to Undefined mode.
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.

PLOVTEN, bit [8]

Traps PL0 accesses to the virtual timer registers to Undefined mode.

PLOVTEN	Meaning
0b0	PL0 accesses to the CNTV_CTL , CNTV_CVAL , and CNTV_TVAL registers are trapped to Undefined mode.
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 the counter register [CNTVCT](#), as seen from EL1, is the trigger for the event stream generated from that counter, when that stream is enabled.

If FEAT_ECV is implemented, and CNTKCTL.EVENTIS is 1, this field selects a trigger bit in the range 8 to 23 of the counter register [CNTVCT](#).

Otherwise, this field selects a trigger bit in the range 0 to 15 of the counter register [CNTVCT](#).

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 counter register [CNTVCT](#) trigger bit, as seen from EL1 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 the counter register [CNTVCT](#) as seen from EL1.

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.

PLOVCTEN, bit [1]

Traps PL0 accesses to the frequency register and virtual counter register to Undefined mode.

PLOVCTEN	Meaning
0b0	PL0 accesses to the CNTVCT are trapped to Undefined mode. PL0 accesses to the CNTFRQ register are trapped to Undefined mode, if CNTKCTL .PLOPCTEN 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.

PLOPCTEN, bit [0]

Traps PL0 accesses to the frequency register and physical counter register to Undefined mode.

PLOPCTEN	Meaning
0b0	PL0 accesses to the CNTPCT are trapped to Undefined mode. PL0 accesses to the CNTFRQ register are trapped to Undefined mode, if CNTKCTL .PLOVCTEN 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.

Accessing CNTKCTL

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	0b0001	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    return CNTKCTL;
elsif PSTATE.EL == EL2 then
    return CNTKCTL;
elsif PSTATE.EL == EL3 then
    return CNTKCTL;
```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b0001	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    CNTKCTL = R[t];
elsif PSTATE.EL == EL2 then
    CNTKCTL = R[t];
elsif PSTATE.EL == EL3 then
    CNTKCTL = R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTP_TVAL, Counter-timer Physical Timer TimerValue register

The CNTP_TVAL characteristics are:

Purpose

Holds the timer value for the EL1 physical timer.

Configuration

AArch32 System register CNTP_TVAL bits [31:0] are architecturally mapped to AArch64 System register [CNTP_TVAL_EL0\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to CNTP_TVAL are UNDEFINED.

Attributes

CNTP_TVAL 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
TimerValue																															

TimerValue, bits [31:0]

The TimerValue view of the EL1 physical timer.

On a read of this register:

- If [CNTP_CTL.ENABLE](#) is 0, the value returned is UNKNOWN.
- If [CNTP_CTL.ENABLE](#) is 1, the value returned is ([CNTP_CVAL](#) - [CNTPCT](#)).

On a write of this register, [CNTP_CVAL](#) is set to ([CNTPCT](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTP_CTL.ENABLE](#) is 1, the timer condition is met when ([CNTPCT](#) - [CNTP_CVAL](#)) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTP_CTL.ISTATUS](#) is set to 1.
- If [CNTP_CTL.IMASK](#) is 0, an interrupt is generated.

When [CNTP_CTL.ENABLE](#) is 0, the timer condition is not met, but [CNTPCT](#) 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 CNTP_TVAL

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	0b0010	0b000

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0PTEN == '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 == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            return bits(32) UNKNOWN;CNTHPS_TVAL_EL2;
        else
            return (CNTHPS_CVAL_EL2 - PhysicalCountInt())<31:0>;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
        if CNTHP_CTL_EL2.ENABLE == '0' then
            return bits(32) UNKNOWN;CNTHP_TVAL_EL2;
        else
            return (CNTHP_CVAL_EL2 - PhysicalCountInt())<31:0>;
    elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && !ELUsingAArch32(EL2) && SCR_EL3.ECVEn
== '1' && CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11' then
        if CNTP_CTL.ENABLE == '0' then
            return bits(32) UNKNOWN;
        else
            return (CNTP_CVAL - (PhysicalCountInt() - CNTPOFF_EL2))<31:0>;
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) then
        if SCR.NS == '1' then
            if CNTP_CTL_NS.ENABLE == '0' then
                return bits(32) UNKNOWN;CNTP_TVAL;
            else
                return (CNTP_CVAL_NS - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL_S.ENABLE == '0' then
                return bits(32) UNKNOWN;
            else
                return (CNTP_CVAL_S - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL.ENABLE == '0' then
                return bits(32) UNKNOWN;
            else
                return (CNTP_CVAL - PhysicalCountInt())<31:0>;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0'
then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
            AArch32.TakeHypTrapException(0x03);
        elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && !ELUsingAArch32(EL2) && SCR_EL3.ECVEn
== '1' && CNTHCTL_EL2.ECV == '1' then

```



```

    if CNTP_CTL.ENABLE == '0' then
        return bits(32) UNKNOWN;
    else
        return (CNTP_CVAL - (PhysicalCountInt() - CNTPOFF_EL2))<31:0>;
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
        if CNTP_CTL_NS.ENABLE == '0' then
            return bits(32) UNKNOWNCNTP_TVAL_NS;
        else
            return (CNTP_CVAL_NS - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL;
            else
                return (CNTP_CVAL - PhysicalCountInt())<31:0>;
    elsif PSTATE.EL == EL2 then
        if HaveEL(EL3) && ELUsingAArch32(EL3) then
            if CNTP_CTL_NS.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL_NS;
            else
                return (CNTP_CVAL_NS - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL;
            else
                return (CNTP_CVAL - PhysicalCountInt())<31:0>;
    elsif PSTATE.EL == EL3 then
        if SCR.NS == '0' then
            if CNTP_CTL_S.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL_S;
            else
                return (CNTP_CVAL_S - PhysicalCountInt())<31:0>;
        else
            if CNTP_CTL_NS.ENABLE == '0' then
                return bits(32) UNKNOWNCNTP_TVAL_NS;
            else
                return (CNTP_CVAL_NS - PhysicalCountInt())<31:0>;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b0010	0b000

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0PTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0PTEN == '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 == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '10' && CNTHCTL_EL2.EL1PTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && CNTHCTL_EL2.EL0PTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
        CNTHPS_CVAL_EL2CNTHPS_TVAL_EL2 = SignExtend(R[t],64) + PhysicalCountInt();};
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
        CNTHP_CVAL_EL2CNTHP_TVAL_EL2 = SignExtend(R[t],64) + PhysicalCountInt();};
        elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && !ELUsingAArch32(EL2) && SCR_EL3.ECVEn
== '1' && CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11' then
            CNTP_CVAL = SignExtend(R[t],64) + PhysicalCountInt() - CNTPOFF_EL2;
        elseif HaveEL(EL3) && ELUsingAArch32(EL3) then
            if SCR.NS == '1' then
                CNTP_CVAL_NS = SignExtend(R[t],64) + PhysicalCountInt();
            else
                CNTP_CVAL_S = SignExtend(R[t],64) + PhysicalCountInt();
        else
            CNTP_CVALCNTP_TVAL = SignExtend(R[t],64) + PhysicalCountInt();};
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '0' && CNTHCTL_EL2.EL1PCEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.E2H == '1' && CNTHCTL_EL2.EL1PTEN == '0'
then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && CNTHCTL.PL1PCEN == '0' then
        AArch32.TakeHypTrapException(0x03);
    elseif IsFeatureImplemented(FEAT_ECV) && EL2Enabled() && !ELUsingAArch32(EL2) && SCR_EL3.ECVEn
== '1' && CNTHCTL_EL2.ECV == '1' then
        CNTP_CVAL = SignExtend(R[t],64) + PhysicalCountInt() - CNTPOFF_EL2;
    elseif HaveEL(EL3) && ELUsingAArch32(EL3) then
        CNTP_CVAL_NSCNTP_TVAL_NS = SignExtend(R[t],64) + PhysicalCountInt();};
    else
        CNTP_CVALCNTP_TVAL = SignExtend(R[t],64) + PhysicalCountInt();};
elseif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        CNTP_CVAL_NSCNTP_TVAL_NS = SignExtend(R[t],64) + PhysicalCountInt();};
    else
        CNTP_CVALCNTP_TVAL = SignExtend(R[t],64) + PhysicalCountInt();};
elseif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        CNTP_CVAL_SCNTP_TVAL_S = SignExtend(R[t],64) + PhysicalCountInt();};
    else
        CNTP_CVAL_NSCNTP_TVAL_NS = SignExtend(R[t],64) + PhysicalCountInt();};

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

(old)

htmldiff from-

(new)

CNTV_TVAL, Counter-timer Virtual Timer TimerValue register

The CNTV_TVAL characteristics are:

Purpose

Holds the timer value for the virtual timer.

Configuration

AArch32 System register CNTV_TVAL bits [31:0] are architecturally mapped to AArch64 System register [CNTV_TVAL_EL0\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to CNTV_TVAL are UNDEFINED.

Attributes

CNTV_TVAL 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
TimerValue																															

TimerValue, bits [31:0]

The TimerValue view of the virtual timer.

On a read of this register:

- If [CNTV_CTL.ENABLE](#) is 0, the value returned is UNKNOWN.
- If [CNTV_CTL.ENABLE](#) is 1, the value returned is ([CNTV_CVAL](#) - [CNTVCT](#)).

On a write of this register, [CNTV_CVAL](#) is set to ([CNTVCT](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTP_CTL.ENABLE](#) is 1, the timer condition is met when ([CNTVCT](#) - [CNTP_CVAL](#)) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- [CNTV_CTL.ISTATUS](#) is set to 1.
- If [CNTV_CTL.IMASK](#) is 0, an interrupt is generated.

When [CNTV_CTL.ENABLE](#) is 0, the timer condition is not met, but [CNTVCT](#) 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 CNTV_TVAL

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	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
        if CNTHVS_CTL_EL2.ENABLE == '0' then
            return bits(32) UNKNOWN<CNTHVS_TVAL_EL2>;
        else
            return (CNTHVS_CVAL_EL2 - PhysicalCountInt())<31:0>;
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
        if CNTHV_CTL_EL2.ENABLE == '0' then
            return bits(32) UNKNOWN<CNTHV_TVAL_EL2>;
        else
            return (CNTHV_CVAL_EL2 - PhysicalCountInt())<31:0>;
    else
        if CNTV_CTL.ENABLE == '0' then
            return bits(32) UNKNOWN<CNTV_TVAL>;
        elseif HaveEL(EL2) && !ELUsingAArch32(EL2) then
            return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF_EL2))<31:0>;
        elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
            return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF))<31:0>;
        else
            return (CNTV_CVAL - PhysicalCountInt())<31:0>;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && CNTHCTL_EL2.EL1TVT == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            if CNTV_CTL.ENABLE == '0' then
                return bits(32) UNKNOWN<CNTV_TVAL>;
            elseif HaveEL(EL2) && !ELUsingAArch32(EL2) then
                return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF_EL2))<31:0>;
            elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
                return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF))<31:0>;
            else
                return (CNTV_CVAL - PhysicalCountInt())<31:0>;
    elseif PSTATE.EL == EL2 then
        if CNTV_CTL.ENABLE == '0' then
            return bits(32) UNKNOWN<CNTV_TVAL>;
        else
            return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF))<31:0>;
    elseif PSTATE.EL == EL3 then
        if CNTV_CTL.ENABLE == '0' then
            return bits(32) UNKNOWN<CNTV_TVAL>;
        elseif HaveEL(EL2) then
            return (CNTV_CVAL - (PhysicalCountInt() - CNTVOFF))<31:0>;
        else
            return (CNTV_CVAL - PhysicalCountInt())<31:0>;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b0011	0b000

```

if PSTATE.EL == EL0 then
    if !ELUsingAArch32(EL1) && !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CNTKCTL_EL1.EL0VTEN
== '0' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && CNTKCTL.PL0VTEN == '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' && CNTHCTL_EL2.EL0VTEN
== '0' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> != '11' && CNTHCTL_EL2.EL1TVT
== '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '0'
&& IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2CNTHVS_TVAL_EL2 = SignExtend(R[t],64) + PhysicalCountInt();
    elseif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.<E2H,TGE> == '11' && SCR_EL3.NS == '1'
then
        CNTHV_CVAL_EL2CNTHV_TVAL_EL2 = SignExtend(R[t],64) + PhysicalCountInt();
    else
        ifCNTV_TVAL HaveEL(EL2) && !ELUsingAArch32(EL2) then
            CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt() - CNTVOFF_EL2;
        elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
            CNTV_CVAL = SignExtend(R[t], 64) + PhysicalCountInt() - CNTVOFF;
        else
            CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt();
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && !ELUsingAArch32(EL2) && CNTHCTL_EL2.EL1TVT == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            ifCNTV_TVAL HaveEL(EL2) && !ELUsingAArch32(EL2) then
                CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt() - CNTVOFF_EL2;
            elseif HaveEL(EL2) && ELUsingAArch32(EL2) then
                CNTV_CVAL = SignExtend(R[t], 64) + PhysicalCountInt() - CNTVOFF;
            else
                CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt();
    elseif PSTATE.EL == EL2 then
        CNTV_CVALCNTV_TVAL = SignExtend(R[t], 64) + PhysicalCountInt() - CNTVOFF;
    elseif PSTATE.EL == EL3 then
        ifCNTV_TVAL HaveEL(EL2) then
            CNTV_CVAL = SignExtend(R[t], 64) + PhysicalCountInt() - CNTVOFF;
        else
            CNTV_CVAL = SignExtend(R[t],64) + PhysicalCountInt();

```

3020/09/2021 1412:5237: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTVOFF, Counter-timer Virtual Offset register

The CNTVOFF characteristics are:

Purpose

Holds the 64-bit virtual offset. This is the offset between the physical count value visible in [CNTPCT](#) and the virtual count value visible in [CNTVCT](#).

Configuration

AArch32 System register CNTVOFF bits [63:0] are architecturally mapped to AArch64 System register [CNTVOFF_EL2\[63:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to CNTVOFF are UNDEFINED.

If EL2 is not implemented, this register is RES0 from EL3 and the virtual counter uses a fixed virtual offset of zero.

Note

When EL2 is implemented and is using AArch64, if [HCR_EL2](#).{E2H, TGE} is {1, 1}, the virtual counter uses a fixed virtual offset of zero when [CNTVCT](#) is read from Non-secure EL0.

Attributes

CNTVOFF is 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
Virtual offset																															
31	30	29	28	27	26	25	24	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 offset.

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 CNTVOFF

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

MRRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm>

coproc	CRm	opc1
0b1111	0b1110	0b0100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    return CNTVOFF;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        return CNTVOFF;

```

MCRR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm>

coproc	CRm	opc1
0b1111	0b1110	0b0100

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    CNTVOFF = R[t2]:R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        CNTVOFF = R[t2]:R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbdb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

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.

Cache prefetch predictions determined by the actions of code in the target execution context or contexts appearing in program order before the instruction cannot **influence** ~~exploitatively control~~ speculative execution 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 ASID 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);
    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' && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HFGITR_EL2.CPPRCTX == '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.RestrictPredictionCPPRCTX(R[t], RestrictType_CachePrefetch);});
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.RestrictPredictionCPPRCTX(R[t], RestrictType_CachePrefetch);});
elseif PSTATE.EL == EL2 then
    AArch32.RestrictPredictionCPPRCTX(R[t], RestrictType_CachePrefetch);});
elseif PSTATE.EL == EL3 then
    AArch32.RestrictPredictionCPPRCTX(R[t], RestrictType_CachePrefetch);});

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CPSR, Current Program Status Register

The CPSR characteristics are:

Purpose

Holds PE status and control information.

Configuration

This register is present only when AArch32 is supported. Otherwise, direct accesses to CPSR are UNDEFINED.

Attributes

CPSR 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
N	Z	C	V	Q	RES0	SSBS	PAND	DIT	RES0		GE							RES0				E	A	I	F	RES0	RES1			M	

N, bit [31]

Negative condition flag. Set to bit[31] of the result of the last flag-setting instruction. If the result is regarded as a two's complement signed integer, then N is set to 1 if the result was negative, and N is set to 0 if the result was positive or zero.

Z, bit [30]

Zero condition flag. Set to 1 if the result of the last flag-setting instruction was zero, and to 0 otherwise. A result of zero often indicates an equal result from a comparison.

C, bit [29]

Carry condition flag. Set to 1 if the last flag-setting instruction resulted in a carry condition, for example an unsigned overflow on an addition.

V, bit [28]

Overflow condition flag. Set to 1 if the last flag-setting instruction resulted in an overflow condition, for example a signed overflow on an addition.

Q, bit [27]

Cumulative saturation bit. Set to 1 to indicate that overflow or saturation occurred in some instructions.

Bits [26:24]

Reserved, RES0.

SSBS, bit [23]**When FEAT_SSBS is implemented:**

Speculative Store Bypass Safe.

Prohibits speculative loads or stores that might practically allow a cache timing side channel.

A cache timing side channel might be exploited where a load or store uses an address that is derived from a register that is being loaded from memory using a load instruction speculatively read from a memory location. If PSTATE.SSBS is enabled, the address derived from the load instruction might be from earlier in the coherence order than the latest store to that memory location with the same virtual address.

SSBS	Meaning
0b0	Hardware is not permitted to load or store speculatively in the manner described.
0b1	Hardware is permitted to load or store speculatively in the manner described.

The value of this bit is usually set to the value described by the [SCTLR.DSSBS](#) bit on exceptions to any mode except Hyp mode, and the value described by [H SCTLR.DSSBS](#) on exceptions to Hyp mode.

The reset behavior of this field is:

- On a Warm reset, this field resets to an IMPLEMENTATION DEFINED value.

Otherwise:

Reserved, RES0.

PAN, bit [22]**When FEAT_PAN is implemented:**

Privileged Access Never.

PAN	Meaning
0b0	The translation system is the same as Armv8.0.
0b1	Disables privileged read and write accesses to addresses accessible at EL0.

The value of this bit is usually preserved on taking an exception, except in the following situations:

- When the target of the exception is EL1, and the value of the [SCTLR.SPAN](#) bit for the current Security state is 0, this bit is set to 1.
- When the target of the exception is EL3, from Secure state, and the value of the Secure [SCTLR.SPAN](#) is 0, this bit is set to 1.
- When the target of the exception is EL3, from Non-secure state, this bit is set to 0 regardless of the value of the Secure [SCTLR.SPAN](#) bit.

Otherwise:

Reserved, RES0.

DIT, bit [21]**When FEAT_DIT is implemented:**

Data Independent Timing.

DIT	Meaning
0b0	The architecture makes no statement about the timing properties of any instructions.
0b1	<p>The architecture requires that:</p> <ul style="list-style-type: none"> • The timing of every load and store instruction is insensitive to the value of the data being loaded or stored. • For certain data processing instructions, the instruction takes a time that is independent of: <ul style="list-style-type: none"> ◦ The values of the data supplied in any of its registers. ◦ The values of the NZCV flags. • For certain data processing instructions, the response of the instruction to asynchronous exceptions does not vary based on: <ul style="list-style-type: none"> ◦ The values of the data supplied in any of its registers. ◦ The values of the NZCV flags.

The data processing instructions affected by this bit are:

- All cryptographic instructions. These instructions are:
 - AESD, AESE, AESIMC, AESMC, SHA1C, SHA1H, SHA1M, SHA1P, SHA1SU0, SHA1SU1, SHA256H, SHA256H2, SHA256SU0, and SHA256SU1.
- A subset of the instructions that use the general-purpose register file. For these instructions, the effects of CPSR.DIT apply only if they do not use R15 as either their source or destination and pass their condition execution check. These instructions are:
 - BFI, BFC, CLZ, CMN, CMP, MLA, MLAS, MLS, MOVT, MUL, MULS, NOP, PKHBT, PKHTB, RBIT, REV, REV16, REVSH, RRX, SADD16, SADD8, SASX, SBFX, SHADD16, SHADD8, SHASX, SHSAX, SHSUB16, SHSUB8, SMLAL**, SMLAW*, SMLSD*, SMLLA*, SMLLS*, SMMUL*, SMUAD*, SMUL*, SSAX, SSUB16, SSUB8, SXTAB*, SXTAH, SXTB*, SXTH, TEQ, TST, UADD*, UASX, UBFX, UHADD*, UHASX, UHSAX, UHSUB*, UMAAL, UMLAL, UMLALS, UMULL, UMULLS, USADA8, USAX, USUB*, UXTAB*, UXTAH, UXTB*, UXTH, ADC (register-shifted register), ADCS (register-shifted register), ADD (register-shifted register), ADDS (register-shifted register), AND (register-shifted register), ANDS (register-shifted register), ASR (register-shifted register), ASRS (register-shifted register), BIC (register-shifted register), BICS (register-shifted register), EOR (register-shifted register), EORS (register-shifted register), LSL (register-shifted register), LSLS (register-shifted register), LSR (register-shifted register), LSRS (register-shifted register), MOV (register-shifted register), MOVS (register-shifted register), MVN (register-shifted register), MVNS (register-shifted register), ORR (register-shifted register), ORRS (register-shifted register), ROR (register-shifted register), RORS (register-shifted register), RSB (register-shifted register), RSBS (register-shifted register), RSC (register-shifted register), RSCS (register-shifted register), SBC (register-shifted register), SBCS (register-shifted register), SUB (register-shifted register), and SUBS (register-shifted register).
- A subset of the instructions that use the general-purpose register file. For these instructions, the effects of CPSR.DIT apply only if they do not use R15 as either their source or destination. The effects of CPSR.DIT do not depend on these instructions passing their condition execution check. These instructions are:
 - ADC (immediate), ADC (register), ADCS (immediate), ADCS (register), ADD (immediate), ADD (register), ADDS (immediate), ADDS (register), AND (immediate), AND (register), ANDS (immediate), ANDS (register), ASR (immediate), ASR (register), ASRS (immediate), ASRS (register), BIC (immediate), BIC (register), BICS (immediate), BICS (register), EOR (immediate), EOR (register), EORS (immediate), EORS (register), LSL (immediate), LSL (register), LSLS (immediate), LSLS (register), LSR (immediate), LSR (register), LSRS (immediate), LSRS (register), MOV (immediate), MOV (register), MOVS (immediate), MOVS (register), MVN (immediate), MVN (register), MVNS (immediate), MVNS (register), ORR (immediate), ORR (register), ORRS (immediate), ORRS (register), ROR (immediate), ROR (register), RORS (immediate), RORS (register), RSB (immediate), RSB (register), RSBS (immediate), RSBS (register), RSC (immediate), RSC (register), RSCS (immediate), RSCS (register), SBC (immediate), SBC (register), SBCS (immediate), SBCS (register), SUB (immediate), SUB (register), SUBS (immediate), and SUBS (register).
 - If FEAT_CRC32 is implemented, CRC32B, CRC32H, CRC32W, CRC32CB, CRC32CH, and CRC32CW.
- A subset of the instructions that use the SIMD&FP register file. For these instructions, the effects of CPSR.DIT apply only if they pass their condition execution check. These instructions are:

- VABA*, VABD* (integer), VADD (integer), VADDHN, VADDL, VADDW, VAND, VBIC, VBIF, VBIT, VBSL, VCLS, VCLZ, VCNT, VDUP, VEOR, VEXT, VHADD, VHSUB, VMAX (integer), VMIN (integer), VMLA (integer), VMLAL, VMLS (integer), VMLS, VMOV, VMOVL, VMOVN, VMUL (integer and polynomial), VMULL (integer and polynomial), VMVN, VORN, VORR, VPADAL, VPADD (integer), VPADDL, VPMAX (integer), VPMIN (integer), VRADDHN, VREV*, VRHADD, VRSHL, VRSHR, VRSRN, VRSRA, VRSUBHN, VSHL, VSHLL, VSHR, VSLI, VSRA, VSRI, VSUB (integer), VSUBHN, VSUBL, VSUBW, VSWP, VTBL, VTBX, VTRN, VTST, VUZP, and VZIP.
- Another subset of the instructions that use the SIMD&FP register file. For these instructions, the effects of CPSR.DIT apply only if they pass their condition execution check and apply only when the instructions are operating on integer vector elements. These instructions are:
 - VABS, VCGE, VCGT, VCLE, VCLT, VMLA (by scalar), VMLS (by scalar), and VNEG.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [20]

Reserved, RES0.

GE, bits [19:16]

Greater than or Equal flags, for parallel addition and subtraction.

Bits [15:10]

Reserved, RES0.

E, bit [9]

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0b0	Little-endian operation
0b1	Big-endian operation.

Instruction fetches ignore this bit.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

When the reset value of the SCTLR.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

A, bit [8]

SError interrupt mask bit. The possible values of this bit are:

A	Meaning
0b0	Exception not masked.
0b1	Exception masked.

I, bit [7]

IRQ mask bit. ~~The possible values of this bit are:~~

I	Meaning
0b0	Exception not masked.
0b1	Exception masked.

F, bit [6]

FIQ mask bit. ~~The possible values of this bit are:~~

F	Meaning
0b0	Exception not masked.
0b1	Exception masked.

Bit [5]

Reserved, RES0.

Bit [4]

Reserved, RES1.

M, bits [3:0]

Current PE mode. ~~Possible values are:~~

M	Meaning
0b0000	User.
0b0001	FIQ.
0b0010	IRQ.
0b0011	Supervisor.
0b0110	Monitor.
0b0111	Abort.
0b1010	Hyp.
0b1011	Undefined.
0b1111	System.

Accessing CPSR

CPSR can be read using the MRS instruction and written using the MSR (register) or MSR (immediate) instructions.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbfdb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(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	hSUHD_imp	RES0	SE_imp	RES0											

WRPs, bits [31:28]

The number of watchpoints implemented, minus 1.

Permitted values of this field are from 0b0001 for 2 implemented watchpoints, to 0b1111 for 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]

The number of breakpoints implemented, minus 1.

Permitted values of this field are from 0b0001 for 2 implemented breakpoint, to 0b1111 for 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 number of breakpoints that can be used for Context matching, minus 1.

Permitted values of this field are from 0b0000 for 1 Context matching breakpoint, to 0b1111 for 16 Context matching breakpoints.

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.

Version, bits [19:16]

The Debug architecture version. Indicates presence of Armv8 debug architecture. Defined values are:

Version	Meaning
0b00000b0001	Not Armv6, supported. v6 Debug architecture.
0b00010b0010	Armv6, v6v6.1 Debug architecture, with System registers access. architecture.
0b00100b0011	Armv6Armv7, v6.1v7 Debug architecture, with Systembaseline CP14 registers access. implemented.
0b00110b0100	Armv7, v7 Debug architecture, with onlyall baselineCP14 Systemregisters registers. implemented.
0b01000b0101	Armv7, v7v7.1 Debug architecture, with all System registers implemented. architecture.
0b01010b0110	Armv7Armv8, v7.1v8 Debug architecture, with System registers access. architecture.
0b01100b0111	Armv8Armv8.1, debugv8 architecture. Debug architecture, with Virtualization Host Extensions.
0b01110b1000	Armv8Armv8.2, debugv8.2 architecture Debug with Virtualization Host Extensions. architecture.
0b10000b1001	Armv8.2Armv8.4, debugv8.4 architecture, Debug architecture. FEAT_Debugv8p2.
0b1001	Armv8.4 debug architecture, FEAT_Debugv8p4.
0b1010	Armv8.8 debug architecture, FEAT_Debugv8p8.

All other values are reserved.

TheIn any Armv8 implementation, the values 0b0000, 0b0001, 0b0010, 0b0011, 0b0100, and 0b0101 are not permitted in Armv8. permitted.

- If FEAT_VHE is not implemented, the only permitted value is 0b0110.
- In an Armv8.0 implementation, the value 0b1000 or higher is not permitted.

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.

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.

Bit [15]

Reserved, RES1.

nSUHD_imp, bit [14]

In Armv7-A, was Secure User Halting Debug 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
    return 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
                    return 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
                        return 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
                        return DBGDIDR;
            elsif PSTATE.EL == EL3 then
                return DBGDIDR;

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DBGDSCRext, Debug Status and Control Register, External View

The DBGDSCRext characteristics are:

Purpose

Main control register for the debug implementation.

Configuration

AArch32 System register DBGDSCRext bits [31:0] are architecturally mapped to AArch64 System register [MDSCR_EL1\[31:0\]](#).

AArch32 System register DBGDSCRext bit [15] is architecturally mapped to AArch32 System register [DBGDSCRint\[15\]](#).

AArch32 System register DBGDSCRext bit [12] is architecturally mapped to AArch32 System register [DBGDSCRint\[12\]](#).

AArch32 System register DBGDSCRext bits [5:2] are architecturally mapped to AArch32 System register [DBGDSCRint\[5:2\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to DBGDSCRext are UNDEFINED.

This register is required in all implementations.

Attributes

DBGDSCRext 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
TFO	RXfull	TXfull	RES0	RXO	TXU	RES0	INTdis	TDARE	RES0	SC2	NS	SPNIDdis	SPIDdis	MDBGen	HDE	RES0	UDCCdis	RES0	ERRM	ERRM	ERRM	ERRM	ERRM	ERRM	ERRM	ERRM

TFO, bit [31]

When FEAT_TRF is implemented:

Trace Filter override. Used for save/restore of [EDSCR.TFO](#).

When the OS Lock is unlocked, [DBGOSLSR.OSLK](#) == 0, software must treat this bit as UNK/SBZP.

When the OS Lock is locked, [DBGOSLSR.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 [DBGOSLSR.OSLK](#) == 1, access to this field is **RW**.
- When [DBGOSLSR.OSLK](#) == 0, access to this field is **RO**.

Otherwise:

Reserved, RES0.

RXfull, bit [30]

DTRRX full. Used for save/restore of [EDSCR.RXfull](#).

When [DBGOSLSR.OSLK](#) == 0, software must treat this bit as UNK/SBZP.

When [DBGOSLSR.OSLK](#) == 1, this bit holds the value of [EDSCR.RXfull](#). Reads and writes of this bit are indirect accesses to [EDSCR.RXfull](#).

Arm deprecates use of this bit other than for save/restore. Use [DBGDSCRint](#) to access the DTRRX full status.

The architected behavior of this field determines the value it returns after a reset.

Accessing this field has the following behavior:

- When [DBGOSLSR.OSLK](#) == 1, access to this field is **RW**.
- When [DBGOSLSR.OSLK](#) == 0, access to this field is **RO**.

TXfull, bit [29]

DTRTX full. Used for save/restore of [EDSCR.TXfull](#).

When [DBGOSLSR.OSLK](#) == 0, software must treat this bit as UNK/SBZP.

When [DBGOSLSR.OSLK](#) == 1, this bit holds the value of [EDSCR.TXfull](#). Reads and writes of this bit are indirect accesses to [EDSCR.TXfull](#).

Arm deprecates use of this bit other than for save/restore. Use [DBGDSCRint](#) to access the DTRTX full status.

The architected behavior of this field determines the value it returns after a reset.

Accessing this field has the following behavior:

- When [DBGOSLSR.OSLK](#) == 1, access to this field is **RW**.
- When [DBGOSLSR.OSLK](#) == 0, access to this field is **RO**.

Bit [28]

Reserved, RES0.

RXO, bit [27]

Used for save/restore of [EDSCR.RXO](#).

When [DBGOSLSR.OSLK](#) == 0, software must treat this bit as UNK/SBZP.

When [DBGOSLSR.OSLK](#) == 1, this bit holds the value of [EDSCR.RXO](#). Reads and writes of this bit are indirect accesses to [EDSCR.RXO](#).

When [DBGOSLSR.OSLK](#) == 1, if bits [27,6] of the value written to [DBGDSCRext](#) 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 [DBGOSLSR.OSLK](#) == 1, access to this field is **RW**.
- When [DBGOSLSR.OSLK](#) == 0, access to this field is **RO**.

TXU, bit [26]

Used for save/restore of [EDSCR.TXU](#).

When [DBGOSLSR.OSLK](#) == 0, software must treat this bit as UNK/SBZP.

When [DBGOSLSR.OSLK](#) == 1, this bit holds the value of [EDSCR.TXU](#). Reads and writes of this bit are indirect accesses to [EDSCR.TXU](#).

When [DBGOSLSR.OSLK](#) == 1, if bits [26,6] of the value written to DBGDSCRext 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 [DBGOSLSR.OSLK](#) == 1, access to this field is **RW**.
- When [DBGOSLSR.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 [DBGOSLSR.OSLK](#) == 0, this field is RO, and software must treat it as UNK/SBZP.

When [DBGOSLSR.OSLK](#) == 1, this field is RW and 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 [DBGOSLSR.OSLK](#) == 1, access to this field is **RW**.
- When [DBGOSLSR.OSLK](#) == 0, access to this field is **RO**.

TDA, bit [21]

Used for save/restore of [EDSCR.TDA](#).

When [DBGOSLSR.OSLK](#) == 0, software must treat this bit as UNK/SBZP.

When [DBGOSLSR.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 [DBGOSLSR.OSLK](#) == 1, access to this field is **RW**.
- When [DBGOSLSR.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 [DBGOSLSR.OSLK](#) == 0, software must treat this bit as UNK/SBZP.

When [DBGOSLSR.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 [DBGOSLSR.OSLK](#) == 1, access to this field is **RW**.
- When [DBGOSLSR.OSLK](#) == 0, access to this field is **RO**.

Otherwise:

Reserved, RES0.

NS, bit [18]

Non-secure status.

Arm deprecates use of this field.

NS	Meaning
0b0	Secure state.
0b1	Non-secure state.

Access to this field is **RO**.

SPNIDdis, bit [17]**When EL3 is implemented:**

Secure privileged profiling disabled status bit.

SPNIDdis	Meaning
0b0	Profiling allowed in Secure privileged modes.
0b1	Profiling prohibited in Secure privileged modes.

This field reads as 0 if any of the following applies, and reads as 1 otherwise:

- FEAT_Debugv8p2 is not implemented and ExternalSecureNoninvasiveDebugEnabled() returns TRUE.
- EL3 is using AArch32 and the value of [SDCR.SPME](#) is 1.
- EL3 is using AArch64 and the value of [MDCR_EL3.SPME](#) is 1.

Arm deprecates use of this field.

Access to this field is **RO**.

Otherwise:

Reserved, RES0.

SPIDdis, bit [16]**When EL3 is implemented:**

Secure privileged AArch32 invasive self-hosted debug disabled status bit. The value of this bit depends on the value of [SDCR.SPD](#) and the pseudocode function `AArch32.SelfHostedSecurePrivilegedInvasiveDebugEnabled()`.

SPIDdis	Meaning
0b0	Self-hosted debug enabled in Secure privileged AArch32 modes.
0b1	Self-hosted debug disabled in Secure privileged AArch32 modes.

This bit reads as 1 if any of the following is true and reads as 0 otherwise:

- EL3 is using AArch32 and [SDCR.SPD](#) has the value 0b10.
- EL3 is using AArch64 and [MDCR_EL3.SPD32](#) has the value 0b10.
- EL3 is using AArch32, [SDCR.SPD](#) has the value 0b00, and `AArch32.SelfHostedSecurePrivilegedInvasiveDebugEnabled()` returns FALSE.
- EL3 is using AArch64, [MDCR_EL3.SPD32](#) has the value 0b00, and `AArch32.SelfHostedSecurePrivilegedInvasiveDebugEnabled()` returns FALSE.

Arm deprecates use of this field.

Access to this field is **RO**.

Otherwise:

Reserved, RES0.

MDBGen, bit [15]

Monitor debug events enable. Enable Breakpoint, Watchpoint, and Vector Catch exceptions.

MDBGen	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 0.

HDE, bit [14]

Used for save/restore of [EDSCR.HDE](#).

When [DBGOSLSR.OSLK](#) == 0, software must treat this bit as UNK/SBZP.

When [DBGOSLSR.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 [DBGOSLSR.OSLK](#) == 1, access to this field is **RW**.
- When [DBGOSLSR.OSLK](#) == 0, access to this field is **RO**.

Bit [13]

Reserved, RES0.

UDCCdis, bit [12]

Traps EL0 accesses to the DCC registers to Undefined mode.

UDCCdis	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	EL0 accesses to the DBGDSCRint , DBGDTRRXint , DBGDTRTXint , DBGDIDR , DBGDSAR , and DBGDRAR are trapped to Undefined mode.

Note

All accesses to these registers are trapped, including LDC and STC accesses to [DBGDTRTXint](#) and [DBGDTRRXint](#), and MRRC accesses to [DBGDSAR](#) and [DBGDRAR](#).

Traps of EL0 accesses to the [DBGDTRRXint](#) and [DBGDTRTXint](#) are ignored in Debug state.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Bits [11:7]

Reserved, RES0.

ERR, bit [6]

Used for save/restore of [EDSCR.ERR](#).

When [DBGOSLSR.OSLK](#) == 0, software must treat this bit as UNK/SBZP.

When [DBGOSLSR.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 [DBGOSLSR.OSLK](#) == 1, access to this field is **RW**.
- When [DBGOSLSR.OSLK](#) == 0, access to this field is **RO**.

MOE, bits [5:2]

Method of Entry for debug exception. When a debug exception is taken to an Exception level using AArch32, this field is set to indicate the event that caused the exception:

MOE	Meaning
0b0001	Breakpoint.
0b0011	Software breakpoint (BKPT) instruction.
0b0101	Vector catch.
0b1010	Watchpoint.

The reset behavior of this field is:

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

Bits [1:0]

Reserved, RES0.

Accessing DBGDSCRext

Individual fields within this register might have restricted accessibility when the OS Lock is unlocked, [DBGOSLSR.OSLK](#) == 0. See the field descriptions for more detail.

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	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'" && !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
            return DBGDSCRext;
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
        return DBGDSCRext;
elsif PSTATE.EL == EL3 then
    return DBGDSCRext;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1110	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'" && !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
        DBGDSCRext = 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);
    else
        DBGDSCRext = R[t];
elsif PSTATE.EL == EL3 then
    DBGDSCRext = R[t];

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DBGOSLSR, Debug OS Lock Status Register

The DBGOSLSR characteristics are:

Purpose

Provides status information for the OS Lock.

Configuration

AArch32 System register DBGOSLSR bits [31:0] are architecturally mapped to AArch64 System register [OSLSR_EL1\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to DBGOSLSR are UNDEFINED.

The OS Lock status is also visible in the external debug interface through EDPRSR.

Attributes

DBGOSLSR 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[1]		nTT	OSLK	OSLM[0]						

Bits [31:4]

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 DBGOSLSR[3].
- OSLM[0] is DBGOSLSR[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. ~~The possible values are:~~

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 DBGOSLSR

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	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'" && !ELUsingAArch32(EL3) && MDCR_EL3.TDOSA == '1' then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.<TDE,TDOSA> != '00' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x05);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HDCR.<TDE,TDOSA> != '00' then
        AArch32.TakeHypTrapException(0x05);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDOSA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x05);
        else
            return DBGOSLSR;
    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.TDOSA == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDOSA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.AArch32SystemAccessTrap(EL3, 0x05);
            else
                return DBGOSLSR;
    elsif PSTATE.EL == EL3 then
        return DBGOSLSR;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DCCIMVAC, Data Cache line Clean and Invalidate by VA to PoC

The DCCIMVAC characteristics are:

Purpose

Clean and Invalidate data or unified cache line by virtual address to PoC.

Configuration

AArch32 System instruction DCCIMVAC performs the same function as AArch64 System instruction [DC CIVAC](#).

This instruction is present only when AArch32 is supported. Otherwise, direct accesses to DCCIMVAC are UNDEFINED.

Attributes

DCCIMVAC 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
Virtual address to use																															

Bits [31:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the DCCIMVAC instruction

Execution of this instruction might require an address translation from VA to PA, and that translation might fault.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'AArch32 data cache maintenance instructions (DC*)'.

If FEAT_CMOW is implemented, HCRX_EL2.CMOW is 1, and EL1 or EL0 access is enabled, when executed at EL1 or EL0, the instruction has stage 2 read permission to the VA but does not have stage 2 write permission to the VA, the instruction generates a stage 2 Permission fault.

For more information, see 'AArch32 data cache maintenance instructions (DC*)'.

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	0b1110	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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.TPCP == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TPC == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        AArch32.DC(R[t], CacheOp_CleanInvalidate, CacheOpScope_PoC);
elsif PSTATE.EL == EL2 then
    AArch32.DC(R[t], CacheOp_CleanInvalidate, CacheOpScope_PoC);
elsif PSTATE.EL == EL3 then
    AArch32.DC(R[t], CacheOp_CleanInvalidate, CacheOpScope_PoC);

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

DFSR, Data Fault Status Register

The DFSR characteristics are:

Purpose

Holds status information about the last data fault.

Configuration

AArch32 System register DFSR bits [31:0] are architecturally mapped to AArch64 System register [ESR_EL1\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to DFSR are UNDEFINED.

The current translation table format determines which format of the register is used.

Attributes

DFSR 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
RES0																FnV	AET	CM	Ext	WnR	FS[4]	LPAE	RES0	Domain			FS[3:0]				

Bits [31:17]

Reserved, RES0.

FnV, bit [16]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
0b0	DFAR is valid.
0b1	DFAR is not valid, and holds an UNKNOWN value.

This field is valid only for a synchronous External abort other than a synchronous External abort on a translation table walk. It is RES0 for all other Data Abort exceptions.

The reset behavior of this field is:

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

AET, bits [15:14]

When FEAT_RAS is implemented:

Asynchronous Error Type. When DFSC is 0b010001, describes the PE error state after taking the SError interrupt exception. Possible values are:

AET	Meaning
0b00	Uncontainable (UC).
0b01	Unrecoverable state (UEU).
0b10	Restartable state (UEO).
0b11	Recoverable state (UER).

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other aborts.

In the event of multiple errors 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.

CM, bit [13]

Cache maintenance fault. For synchronous faults, this bit indicates whether a cache maintenance instruction generated the fault. **The possible values of this bit are:**

CM	Meaning
0b0	Abort not caused by execution of a cache maintenance instruction.
0b1	Abort caused by execution of a cache maintenance instruction, or on an address translation.

On a synchronous Data Abort on a translation table walk, this bit is UNKNOWN.

On an asynchronous fault, this bit is UNKNOWN.

The reset behavior of this field is:

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

ExT, bit [12]

External abort type. This bit can be used to provide an IMPLEMENTATION DEFINED classification of External aborts.

In an implementation that does not provide any classification of External aborts, this bit is RES0.

For aborts other than External aborts this bit always returns 0.

The reset behavior of this field is:

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

WnR, bit [11]

Write not Read bit. Indicates whether the abort was caused by a write or a read instruction. **The possible values of this bit are:**

WnR	Meaning
0b0	Abort caused by a read instruction.
0b1	Abort caused by a write instruction.

For faults on the cache maintenance and address translation System instructions in the (coproc==0b1111) encoding space this bit always returns a value of 1.

The reset behavior of this field is:

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

FS, bits [10, 3:0]

Fault status bits. Possible values of FS[4:0] are:

FS	Meaning	Applies when
0b00001	Alignment fault.	
0b00010	Debug exception.	
0b00011	Access flag fault, level 1.	
0b00100	Fault on instruction cache maintenance.	
0b00101	Translation fault, level 1.	
0b00110	Access flag fault, level 2.	
0b00111	Translation fault, level 2.	
0b01000	Synchronous External abort, not on translation table walk.	
0b01001	Domain fault, level 1.	
0b01011	Domain fault, level 2.	
0b01100	Synchronous External abort, on translation table walk, level 1.	
0b01101	Permission fault, level 1.	
0b01110	Synchronous External abort, on translation table walk, level 2.	
0b01111	Permission fault, level 2.	
0b10000	TLB conflict abort.	
0b10100	IMPLEMENTATION DEFINED fault (Lockdown fault).	
0b10101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive access fault).	
0b10110	SError interrupt.	
0b11000	SError interrupt, from a parity or ECC error on memory access.	When FEAT_RAS is not implemented
0b11001	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b11100	Synchronous parity or ECC error on translation table walk, level 1.	When FEAT_RAS is not implemented
0b11110	Synchronous parity or ECC error on translation table walk, level 2.	When FEAT_RAS is not implemented

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults on a Short-descriptor translation table lookup'.

The FS field is split as follows:

- FS[4] is DFSR[10].
- FS[3:0] is DFSR[3:0].

The reset behavior of this field is:

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

LPAE, bit [9]

On taking a Data Abort exception, this bit is set as follows:

LPAA	Meaning
0b0	Using the Short-descriptor translation table formats.
0b1	Using the Long-descriptor translation table formats.

Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.

The reset behavior of this field is:

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

Bit [8]

Reserved, RES0.

Domain, bits [7:4]

The domain of the fault address.

Arm deprecates any use of this field, see 'The Domain field in the DFSR'.

This field is UNKNOWN for certain faults where the DFSR is updated and reported using the Short-descriptor FSR encodings, see 'Validity of Domain field on faults that update the DFSR when using the Short-descriptor encodings'.

The reset behavior of this field is:

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

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
RES0																FnV	AET	CM	Ext	WnR	RES0	LPAA	RES0	STATUS							

Bits [31:17]

Reserved, RES0.

FnV, bit [16]

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
0b0	DFAR is valid.
0b1	DFAR is not valid, and holds an UNKNOWN value.

This field is valid only for a synchronous External abort other than a synchronous External abort on a translation table walk. It is RES0 for all other Data Abort exceptions.

The reset behavior of this field is:

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

AET, bits [15:14]

When FEAT_RAS is implemented:

Asynchronous Error Type. When DFSC is 0b010001, describes the PE error state after taking the SError interrupt exception. Possible values are:

AET	Meaning
0b00	Uncontainable (UC).
0b01	Unrecoverable state (UEU).
0b10	Restartable state (UEO).
0b11	Recoverable state (UER).

This field is valid only if the DFSC code is 0b010001. It is RES0 for all other aborts.

In the event of multiple errors 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.

CM, bit [13]

Cache maintenance fault. For synchronous faults, this bit indicates whether a cache maintenance instruction generated the fault. **The possible values of this bit are:**

CM	Meaning
0b0	Abort not caused by execution of a cache maintenance instruction.
0b1	Abort caused by execution of a cache maintenance instruction.

On a synchronous Data Abort on a translation table walk, this bit is UNKNOWN.

On an asynchronous fault, this bit is UNKNOWN.

The reset behavior of this field is:

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

ExT, bit [12]

External abort type. This bit can be used to provide an IMPLEMENTATION DEFINED classification of External aborts.

In an implementation that does not provide any classification of External aborts, this bit is RES0.

For aborts other than External aborts this bit always returns 0.

The reset behavior of this field is:

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

WnR, bit [11]

Write not Read bit. Indicates whether the abort was caused by a write or a read instruction. **The possible values of this bit are:**

WnR	Meaning
0b0	Abort caused by a read instruction.
0b1	Abort caused by a write instruction.

For faults on the cache maintenance and address translation System instructions in the (coproc==0b1111) encoding space this bit always returns a value of 1.

The reset behavior of this field is:

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

Bit [10]

Reserved, RES0.

LPAE, bit [9]

On taking a Data Abort exception, this bit is set as follows:

LPAE	Meaning
0b0	Using the Short-descriptor translation table formats.
0b1	Using the Long-descriptor translation table formats.

Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.

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 bits. Possible values of this field are:

STATUS	Meaning	Applies when
0b000000	Address size fault in translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
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.	
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk.	
0b010001	Asynchronous SError interrupt.	
0b010101	Synchronous External abort on translation table walk, level 1.	
0b010110	Synchronous External abort on translation table walk, level 2.	
0b010111	Synchronous External abort on translation table walk, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011001	Asynchronous SError interrupt, from a parity or ECC error on memory access.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk, level 3.	When FEAT_RAS is not implemented
0b100001	Alignment fault.	
0b100010	Debug exception.	
0b110000	TLB conflict abort.	
0b110100	IMPLEMENTATION DEFINED fault (Lockdown).	
0b110101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive access).	

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults on a Long-descriptor translation table lookup'.

The reset behavior of this field is:

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

Accessing DFSR

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	0b0101	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T5 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T5 == '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
        return DFSR_NS;
    else
        return DFSR;
elsif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        return DFSR_NS;
    else
        return DFSR;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        return DFSR_S;
    else
        return DFSR_NS;

```

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0101	0b0000	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T5 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T5 == '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
        DFSR_NS = R[t];
    else
        DFSR = R[t];
elsif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        DFSR_NS = R[t];
    else
        DFSR = R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        DFSR_S = R[t];
    else
        DFSR_NS = R[t];

```

DFSR, Data Fault Status Register

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

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				GVMID	NS	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 ASID 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_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' && (!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_EL2.EnRCTX ==
'0' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch32.RestrictPredictionDVPRCTX(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_EL2.T7 == '1' then
            AArch32.TakeHypTrapException(0x03);
        elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.NV == '1' then
            AArch64.SystemAccessTrap(EL2, 0x03);
        else
            AArch32.RestrictPredictionDVPRCTX(R[t], RestrictType_DataValue);});
    elsif PSTATE.EL == EL2 then
        AArch32.RestrictPredictionDVPRCTX(R[t], RestrictType_DataValue);});
    elsif PSTATE.EL == EL3 then
        AArch32.RestrictPredictionDVPRCTX(R[t], RestrictType_DataValue);});

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

HCR, Hyp Configuration Register

The HCR characteristics are:

Purpose

Provides configuration controls for virtualization, including defining whether various Non-secure operations are trapped to Hyp mode.

Configuration

AArch32 System register HCR bits [31:0] are architecturally mapped to AArch64 System register [HCR_EL2\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to HCR are UNDEFINED.

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

Attributes

HCR 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
RES0	TRVM	HCD	RES0	TGE	TVM	TTLB	TPU	TPC	TSW	TACT	TIDCP	TSCT	TID3	TID2	TID1	TID0	TWET	TWI	DCBSU	FB	VAVI	VFA	MOI	MC			

Bit [31]

Reserved, RES0.

TRVM, bit [30]

Trap Reads of Virtual Memory controls. Traps Non-secure EL1 reads of the virtual memory control registers to EL2, when EL2 is enabled in the current Security state.

The registers for which read accesses are trapped are as follows:

[SCTLR](#), [TTBR0](#), [TTBR1](#), [TTBCR](#), [TTBCR2](#), [DACR](#), [DFSR](#), [IFSR](#), [DFAR](#), [IFAR](#), [ADFSR](#), [AIFSR](#), [PRRR](#), [NMRR](#), [MAIR0](#), [MAIR1](#), [AMAIRO](#), [AMAIRO](#), [CONTEXTIDR](#).

TRVM	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL1 read accesses to the specified Virtual Memory controls are trapped to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

HCD, bit [29]

When EL3 is not implemented:

HVC instruction disable. Disables Non-secure EL1 and EL2 execution of HVC instructions, when EL2 is enabled in the current Security state.

HCD	Meaning
0b0	HVC instruction execution is enabled at EL2 and EL1.
0b1	HVC instructions are UNDEFINED at EL2 and Non-secure EL1. The Undefined Instruction 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, in a system where the PE resets into EL2 or EL3, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [28]

Reserved, RES0.

TGE, bit [27]

Trap General Exceptions, from Non-secure EL0.

TGE	Meaning
0b0	This control has no effect on execution at EL0.
0b1	When EL2 is not enabled in the current Security state, this control has no effect on execution at EL0. When EL2 is enabled in the current Security state, then: <ul style="list-style-type: none"> All exceptions that would be routed to EL1 are routed to EL2. The SCTLR.M bit is treated as being 0 for all purposes other than returning the result of a direct read of SCTLR. The HCR.{FMO, IMO, AMO} bits are treated as being 1 for all purposes other than returning the result of a direct read of HCR. 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. Monitor mode execution of an MSR or CPS instruction that changes PSTATE.M to a Non-secure EL1 mode is an illegal change to PSTATE.M. For more information see 'Illegal changes to PSTATE.M'.

Also, when HCR.TGE is 1:

- If EL3 is using AArch32, an attempt to change from a Secure PL1 mode to a Non-secure EL1 mode by changing [SCR.NS](#) from 0 to 1 results in [SCR.NS](#) remaining as 0.
- The [HDCR](#).{TDRA, TDOSA, TDA, TDE} bits are ignored and treated as being 1 other than for the purpose of a direct read of [HDCR](#).

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TVM, bit [26]

Trap Virtual Memory controls. Traps Non-secure EL1 writes to the virtual memory control registers to EL2, when EL2 is enabled in the current Security state.

The registers for which write accesses are trapped are as follows:

[SCTLR](#), [TTBR0](#), [TTBR1](#), [TTBCR](#), [TTBCR2](#), [DACR](#), [DFSR](#), [IFSR](#), [DFAR](#), [IFAR](#), [ADFSR](#), [AIFSR](#), [PRRR](#), [NMRR](#), [MAIRO](#), [MAIR1](#), [AMAIRO](#), [AMAIR1](#), [CONTEXTIDR](#).

TVM	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL1 write accesses to the specified virtual memory control registers are trapped to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TTLB, bit [25]

Trap TLB maintenance instructions. Traps Non-secure EL1 execution of a TLBI instruction to EL2, when EL2 is enabled in the current Security state.

This applies to the following instructions:

[TLBIALLIS](#), [TLBIMVAIS](#), [TLBIASIDIS](#), [TLBIMVAAIS](#), [TLBIMVALIS](#), [TLBIMVAALIS](#), [ITLBIALL](#), [ITLBIMVA](#), [ITLBIASID](#), [DTLBIALL](#), [DTLBIMVA](#), [DTLBIASID](#), [TLBIALL](#), [TLBIMVA](#), [TLBIASID](#), [TLBIMVAA](#), [TLBIMVAL](#), [TLBIMVAAL](#)

TTLB	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL1 accesses to the specified TLB maintenance instructions are trapped to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TPU, bit [24]

Trap cache maintenance instructions that operate to the Point of Unification. Traps Non-secure EL1 execution of those cache maintenance instructions to EL2, when EL2 is enabled in the current Security state.

This applies to the following instructions:

- [ICIMVAU](#), [ICIALLU](#), [ICIALLUIS](#), [DCCMVAU](#).

Note

An Undefined Instruction exception generated at EL0 is higher priority than this trap to EL2, and these instructions are always UNDEFINED at EL0.

TPU	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL1 execution of the specified cache maintenance instructions is 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.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TPC, bit [23]

Trap data or unified cache maintenance instructions that operate to the Point of Coherency. Traps Non-secure EL1 execution of those cache maintenance instructions to EL2, when EL2 is enabled in the current Security state.

This applies to the following instructions:

- [DCIMVAC](#), [DCCIMVAC](#), [DCCMVAC](#).

Note

An Undefined Instruction exception generated at EL0 is higher priority than this trap to EL2, and these instructions are always UNDEFINED at EL0.

TPC	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL1 execution of the specified cache maintenance instructions is trapped to EL2.

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.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TSW, bit [22]

Trap data or unified cache maintenance instructions that operate by Set/Way. Traps Non-secure EL1 execution of those cache maintenance instructions by set/way to EL2, when EL2 is enabled in the current Security state.

This applies to the following instructions:

- [DCISW](#), [DCCSW](#), [DCCISW](#).

Note

An Undefined Instruction exception generated 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	Non-secure EL1 execution of the specified cache maintenance instructions is trapped to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TAC, bit [21]

Trap Auxiliary Control Registers. Traps Non-secure EL1 accesses to the Auxiliary Control Registers to EL2, when EL2 is enabled in the current Security state, from both Execution states.

This applies to the following register accesses:

[ACTLR](#) and, if implemented, [ACTLR2](#).

TAC	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL1 accesses to the specified registers are trapped to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TIDCP, bit [20]

Trap IMPLEMENTATION DEFINED functionality. Traps Non-secure EL1 accesses to the encodings for IMPLEMENTATION DEFINED System Registers to EL2, when EL2 is enabled in the current Security state.

MCR and MRC instructions accessing the following encodings:

- All coproc==p15, CRn==c9, Opcode1 = {0-7}, CRm == {c0-c2, c5-c8}, opcode2 == {0-7}.
- All coproc==p15, CRn==c10, Opcode1 =={0-7}, CRm == {c0, c1, c4, c8}, opcode2 == {0-7}.
- All coproc==p15, CRn==c11, Opcode1=={0-7}, CRm == {c0-c8, c15}, opcode2 == {0-7}.

When HCR.TIDCP is set to 1, it is IMPLEMENTATION DEFINED whether any of this functionality accessed from Non-secure EL0 is trapped to EL2. Otherwise, it is UNDEFINED and the PE takes an Undefined Instruction exception to Non-secure Undefined mode.

TIDCP	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL1 accesses to the specified System register encodings for IMPLEMENTATION DEFINED functionality are trapped to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TSC, bit [19]

Trap SMC instructions. Traps Non-secure EL1 execution of SMC instructions to Hyp mode.

TSC	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any attempt to execute an SMC instruction at Non-secure EL1 is trapped to Hyp mode, regardless of the value of SCR.SCD .

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.

Note

- This trap is only implemented if the implementation includes EL3.
- SMC instructions are always UNDEFINED at PL0.
- This bit traps execution of the SMC instruction. It is not a routing control for the SMC exception. Hyp Trap exceptions and SMC exceptions have different preferred return addresses.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TID3, bit [18]

Trap ID group 3. Traps Non-secure EL1 reads of the following registers to EL2, when EL2 is enabled in the current Security state as follows:

- VMRS access to [MVFR0](#), [MVFR1](#), and [MVFR2](#), reported using EC syndrome value 0x08, unless access is also trapped by [HCPTR](#) which takes priority.
- MRC access to the following registers are 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](#), and [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 registers 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 MRC accesses to registers not already mentioned, 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 Non-secure EL1 read accesses to ID group 3 registers are trapped to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TID2, bit [17]

Trap ID group 2. Traps the following register accesses to EL2, when EL2 is enabled in the current Security state:

- Non-secure EL1 and EL0 reads of the [CTR](#), [CCSIDR](#), [CCSIDR2](#), [CLIDR](#), and [CSSELR](#).
- Non-secure EL1 and EL0 writes to the [CSSELR](#).

TID2	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	The specified Non-secure EL1 and EL0 accesses to ID group 2 registers are trapped to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TID1, bit [16]

Trap ID group 1. Traps Non-secure EL1 reads of the following registers to EL2, when EL2 is enabled in the current Security state:

[TCMTR](#), [TLBTR](#), [REVIDR](#), [AIDR](#).

TID1	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	The specified Non-secure EL1 read accesses to ID group 1 registers are trapped to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TID0, bit [15]

Trap ID group 0. Traps the following register accesses to EL2, when EL2 is enabled in the current Security state:

- Non-secure EL1 reads of the [JIDR](#) and [FPSID](#).
- If the [JIDR](#) is RAZ from Non-secure EL0, Non-secure EL0 reads of the [JIDR](#).

Note

- It is IMPLEMENTATION DEFINED whether the [JIDR](#) is RAZ or UNDEFINED at EL0. If it is UNDEFINED at EL0 then the Undefined Instruction exception takes precedence over this trap.
- The [FPSID](#) is not accessible at EL0.
- 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 Non-secure EL1 read accesses to ID group 0 registers are trapped to EL2.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TWE, bit [14]

Traps Non-secure EL0 and EL1 execution of WFE instructions to EL2, when EL2 is enabled in the current Security state.

TWE	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any attempt to execute a WFE instruction at Non-secure EL0 or EL1 is trapped to EL2, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by SCTLR.nTWE .

The attempted execution of a conditional WFE instruction is only trapped 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.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TWI, bit [13]

Traps Non-secure EL0 and EL1 execution of WFI instructions to EL2, when EL2 is enabled in the current Security state.

TWI	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any attempt to execute a WFI instruction at Non-secure EL0 or EL1 is trapped to EL2, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by SCTLR.nTWI .

The attempted execution of a conditional WFI instruction is only trapped 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.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

DC, bit [12]

Default Cacheability.

DC	Meaning
0b0	This control has no effect on the Non-secure EL1&0 translation regime.
0b1	In Non-secure state: <ul style="list-style-type: none"> The SCTLR.M field behaves as 0 for all purposes other than a direct read of the value of the field. The HCR.VM field behaves as 1 for all purposes other than a direct read of the value of the field. The memory type produced by the first stage 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 and EL3 translation regimes.

This **bitfield** 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 or EL3, this field resets to 0.

BSU, bits [11:10]

Barrier Shareability upgrade. This field determines the minimum shareability domain that is applied to any barrier instruction executed from Non-secure EL1 or Non-secure 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.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

FB, bit [9]

Force broadcast. Causes the following instructions to be broadcast within the Inner Shareable domain when executed from Non-secure EL1:

[BPIALL](#), [TLBIALL](#), [TLBIMVA](#), [TLBIASID](#), [DTLBIALL](#), [DTLBIMVA](#), [DTLBIASID](#), [ITLBIALL](#), [ITLBIMVA](#), [ITLBIASID](#), [TLBIMVAA](#), [ICIALLU](#), [TLBIMVAL](#), [TLBIMVAAL](#).

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 Non-secure EL1, the instruction is broadcast within the Inner Shareable shareability domain.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

VA, bit [8]

Virtual SError interrupt exception.

VA	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.{TGE, AMO} is {0, 1}.

The Guest OS cannot distinguish the virtual exception from the corresponding physical exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

VI, bit [7]

Virtual IRQ exception.

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.{TGE, IMO} is {0, 1}.

The Guest OS cannot distinguish the virtual exception from the corresponding physical exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

VF, bit [6]

Virtual FIQ exception.

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.{TGE, FMO} is {0, 1}.

The Guest OS cannot distinguish the virtual exception from the corresponding physical exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

AMO, bit [5]

SErrors interrupt Mask Override. When this bit is set to 1, it overrides the effect of PSTATE.A, and enables virtual exception signaling by the VA bit.

If the value of HCR.TGE is 0, then virtual SError interrupts are enabled in Non-secure state.

If the value of HCR.TGE is 1, then in Non-secure state the HCR.AMO bit behaves as 1 for all purposes other than a direct read of the value of the bit.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

IMO, bit [4]

IRQ Mask Override. When this bit is set to 1, it overrides the effect of PSTATE.I, and enables virtual exception signaling by the VI bit.

If the value of HCR.TGE is 0, then Virtual IRQ interrupts are enabled in the Non-secure state.

If the value of HCR.TGE is 1, then in Non-secure state the HCR.IMO bit behaves as 1 for all purposes other than a direct read of the value of the bit.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

FMO, bit [3]

FIQ Mask Override. When this bit is set to 1, it overrides the effect of PSTATE.F, and enables virtual exception signaling by the VF bit.

If the value of HCR.TGE is 0, then Virtual FIQ interrupts are enabled in the Non-secure state.

If the value of HCR.TGE is 1, then in Non-secure state the HCR.FMO bit behaves as 1 for all purposes other than a direct read of the value of the bit.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

PTW, bit [2]

Protected Table Walk. In the Non-secure PL1&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 **bitfield** 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 or EL3, this field resets to 0.

SWIO, bit [1]

Set/Way Invalidation Override. Causes Non-secure 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 this bit is set to 1, [DCISW](#) performs the same invalidation as a [DCCISW](#) instruction.

As a result of changes to the behavior of [DCISW](#), this bit is redundant in Armv8. This bit can be implemented as RES1.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

VM, bit [0]

Virtualization enable. Enables stage 2 address translation for the Non-secure EL1&0 translation regime.

VM	Meaning
0b0	Non-secure EL1&0 stage 2 address translation disabled.
0b1	Non-secure EL1&0 stage 2 address translation enabled.

If the HCR.DC bit is set to 1, then the behavior of the PE when executing in a Non-secure mode other than Hyp mode is consistent with HCR.VM being 1, regardless of the actual value of HCR.VM, other than the value returned by an explicit read of HCR.VM.

When the value of this bit is 1, data cache invalidate instructions executed at Non-secure 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.SWIO bit.

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 or EL3, this field resets to 0.

Accessing HCR

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	0b100	0b0001	0b0001	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);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return HCR;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        return HCR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b100	0b0001	0b0001	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);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HCR = R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        HCR = R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

HDCR, Hyp Debug Control Register

The HDCR characteristics are:

Purpose

Controls the trapping to Hyp mode of Non-secure accesses, at EL1 or lower, to functions provided by the debug and trace architectures and the Performance Monitors Extension.

Configuration

AArch32 System register HDCR bits [31:0] are architecturally mapped to AArch64 System register [MDCR_EL2\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to HDCR are UNDEFINED.

If EL2 is not implemented, this register is RES0 from EL3, and other than for a direct read of the register, the PE behaves as if `HDCR.HPMN == PMCR.N`.

Attributes

HDCR 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
RES0	HPMFZO	MTPME	TDCC	HLP	RES0	HCCD	RES0	TTRF	RES0	HPMD	RES0	TDRAT	DOSAT	TDA	TDE	HPMET	TPM	TPMCR	N								

Bits [31:30]

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 PMOVSr [(PMCR.N -1):HDCR.HPMN] is nonzero.

If `HDCR.HPMN` is less than [PMCR.N](#), this field affects the operation of event counters in the range [`HDCR.HPMN` .. ([PMCR.N](#)-1)].

This field does not affect the operation of event counters in the range [0 .. (`HDCR.HPMN`-1)] and [PMCCNTR](#).

The operation of this field ignores the values of [PMOVSr](#)[(`HDCR.HPMN`-1):0].

This field does not affect the operation of other event counters and [PMCCNTR](#). [PMCR.N](#), this field has no effect.

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, in a system where the PE resets into EL2 or EL3, 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>](#).MT bits.

MTPME	Meaning
0b0	FEAT_MTPMU is disabled. The Effective value of PMEVTYPER<n> .MT is zero.
0b1	PMEVTYPER<n> .MT bits not affected by this bit.

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 bit is 0b0.

The reset behavior of this field is:

- On a Cold reset, in a system where the PE resets into EL2 or EL3, 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 Hyp Trap exception, 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:

- [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 MRC and MCR accesses with coproc == 0b1110.
- 0x06 for trapped LDC to [DBGDTRTXint](#) and STC from [DBGDTRRXint](#).

When the PE is in Debug state, HDCR.TDCC does not trap any accesses to:

- [DBGDTRRXint](#) and [DBGDTRTXint](#).

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

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> [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 read/write or RAZ/WI.

If HDCR.HPMN is less than PMCR.N, this bit affects the operation of event counters in the range [HDCR.HPMN..[\(PMCR.N-1\)](#)]. ~~Otherwise this bit has no effect on the operation of the event counters.~~

Note

~~The effect of HDCR.HPMN on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state.~~

~~For more information see the description of the HDCR.HPMN field.~~

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.

Note

[PMEVCNTR<n>](#)[63:32] cannot be accessed directly in AArch32 state.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

Bits [25:24]

Reserved, RES0.

HCCD, bit [23]**When FEAT_PMUv3p5 is implemented:**

Hypervisor Cycle Counter Disable. Prohibits [PMCCNTR](#) from counting at EL2.

HCCD	Meaning
0b0	Cycle counting by PMCCNTR is not affected by this mechanism.
0b1	Cycle counting by PMCCNTR 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, in a system where the PE resets into EL2 or EL3, 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.

TTRF	Meaning
0b0	Accesses to TRFCR at EL1 are not affected by this control bit.
0b1	Accesses to TRFCR at EL1 generate a Hyp Trap exception.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

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 are not affected by this mechanism.
0b1	Event counting by some event counters is prohibited in Hyp mode. If PMCR.DP is 1, PMCCNTR is disabled in Hyp mode. Otherwise, PMCCNTR is not affected by this mechanism.

If this HDCR.HPMN is not 0, this field affects ~~applies~~ the only operation of event counters in the range [0 .. (HDCR.HPMN-1)]. ~~to:~~

- The event counters in the range [0 .. (HDCR.HPMN-1)].
- If [PMCR.DP](#) is 1, [PMCCNTR](#).

~~The other event counters are not affected. When [PMCR.DP](#) is 0, [PMCCNTR](#) is not affected.~~

This field does not affect the operation of other event counters.

If [PMCR.DP](#) is 1, this field affects [PMCCNTR](#).

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, 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 are not affected by this mechanism.
0b1	If <code>ExternalSecureNoninvasiveDebugEnabled()</code> is FALSE, event counting by some event counters is prohibited in Hyp mode, and if PMCR.DP is 1, PMCCNTR is disabled in Hyp mode.

If `ExternalSecureNoninvasiveDebugEnabled()` is TRUE, this field does not affect the event counters and does not affect [PMCCNTR](#). ~~are not affected by this field.~~

Otherwise, this field applies only to:

- If [HDCR.HPMN](#) is not 0, this field affects the operation of ~~The~~ event counters in the range [0 .. ([HDCR.HPMN](#)-1)].
- This field does not affect the operation of other event counters. ~~If [PMCR.DP](#) is 1, [PMCCNTR](#).~~
- If [PMCR.DP](#) is 1, this field affects [PMCCNTR](#).

~~The other event counters are not affected. When [PMCR.DP](#) is 0, [PMCCNTR](#) is not affected.~~

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

Otherwise:

Reserved, RES0.

Bits [16:12]

Reserved, RES0.

TDRA, bit [11]

Trap Debug ROM Address register access. Traps Non-secure EL0 and EL1 System register accesses to the Debug ROM registers to Hyp mode.

TDRA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL0 and EL1 System register accesses to the DBGDRAR or DBGDSAR are trapped to Hyp mode, unless it is trapped by DBGDSCRext .UDCCdis.

If [HCR.TGE](#) or [HDCR.TDE](#) is 1, 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, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TDOSA, bit [10]

When `FEAT_DoubleLock` is implemented:

Trap debug OS-related register access. Traps Non-secure EL1 System register accesses to the powerdown debug registers to Hyp mode.

TDOSA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL1 System register accesses to the powerdown debug registers are trapped to Hyp mode.

The registers for which accesses are trapped are as follows:

- [DBGOSLSR](#), [DBGOSLAR](#), [DBGOSDLR](#), and [DBGPRCR](#).
- Any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit.

Note

These registers are not accessible at EL0.

If [HCR.TGE](#) or [HDCR.TDE](#) is 1, 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, in a system where the PE resets into EL2 or EL3, this field resets to 0.

Otherwise:

Trap debug OS-related register access. Traps Non-secure EL1 System register accesses to the powerdown debug registers to Hyp mode.

TDOSA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL1 System register accesses to the powerdown debug registers are trapped to Hyp mode.

The registers for which accesses are trapped are as follows:

- [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 [DBGOSDLR](#) are trapped.

Note

These registers are not accessible at EL0.

If [HCR.TGE](#) or [HDCR.TDE](#) is 1, 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, in a system where the PE resets into EL2 or EL3, this field resets to 0.

TDA, bit [9]

Trap debug access. Traps Non-secure EL0 and EL1 System register accesses to those debug System registers in the (coproc==0b1110) encoding space that are not trapped by either of the following:

- [HDCR.TDRA](#).
- [HDCR.TDOSA](#).

TDA	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL0 or EL1 System register accesses to the debug registers, other than the registers trapped by HDCR.TDRA and HDCR.TDOSA , are trapped to Hyp mode, unless it is trapped by DBGDSCRext.UDCCdis .

Traps of AArch32 accesses to [DBGDTRRXint](#) and [DBGDTRTXint](#) are ignored in Debug state.

If [HCR.TGE](#) or [HDCR.TDE](#) is 1, 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, in a system where the PE resets into EL2 or EL3, this field resets to 0.

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.NS , the debug target Exception level is EL2, otherwise the debug target Exception level is EL1. The HDCR.{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'.

When [HCR.TGE](#) == 1, the PE behaves as if the value of this field is 1 for all purposes other than returning the value of a direct read of the register.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to 0.

HPME, bit [7]

When [FEAT_PMuV3](#) is implemented:

[HDCR.HPMN..(N-1)] event counters enable.

HPME	Meaning
0b0	Event counters in the range [HDCR.HPMN.. (PMCR.N-1)] are disabled.
0b1	Event counters in the range [HDCR.HPMN.. (PMCR.N-1)] are enabled by PMCNTENSET .

If HDCR.HPMN is less than [PMCR.N](#), this field affects the operation of event counters in the range [HDCR.HPMN..[\(PMCR.N-1\)](#)], are enabled and disabled by this bit. Otherwise this bit has no effect on the operation of the event counters.

Note

The effect of HDCR.HPMN on the operation of this bit applies regardless of whether EL2 is enabled in the current Security state.

For more information see the description of the HPMN field.

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, in a system where the PE resets into EL2 or EL3, this field resets to an architecturally UNKNOWN value.

Otherwise:

Reserved, RES0.

TPM, bit [6]

When [FEAT_PMuV3](#) is implemented:

Trap Performance Monitors accesses. Traps Non-secure EL0 and EL1 accesses to all Performance Monitors registers to Hyp mode.

TPM	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL0 and EL1 accesses to all Performance Monitors registers are trapped to Hyp mode.

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, in a system where the PE resets into EL2 or EL3, this field resets to 0.

Otherwise:

Reserved, RES0.

TPMCR, bit [5]

When FEAT_PMUv3 is implemented:

Trap [PMCR](#) accesses. Traps Non-secure EL0 and EL1 accesses to the [PMCR](#) to Hyp mode.

TPMCR	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Non-secure EL0 and EL1 accesses to the PMCR are trapped to Hyp mode, unless it is trapped by PMUSERENR.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, in a system where the PE resets into EL2 or EL3, this field resets to 0.

Otherwise:

Reserved, RES0.

HPMN, bits [4:0]

When FEAT_PMUv3 is implemented:

Defines the number of event counters that are accessible from Non-secure EL1 modes, and from Non-secure EL0 modes if unprivileged access is enabled.

If HPMN is not 0 and is less than [PMCR.N](#), HPMN divides the event counters into two first ranges, range [0..(HPMN-1)], and a second range [HPMN..([PMCR.N](#)-1)].

If for an event counter in the range [0..(HPMN-1)]: 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.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, and EL2, EL3, and from EL0 if unprivileged access to the counters is enabled.
- The counter is accessible from EL0 if permitted by [PMUSERENR](#).
- If FEAT_PMUv3p5 is implemented, [PMCR.LP](#) determines whether the counter overflows at [PMEVCNTR<n>\[31:0\]](#) or [PMEVCNTR<n>\[63:0\]](#).
- [PMCR.E](#) enables the operation of counters in this range; [PMCNTENSET\[n\]](#) enable the operation of event counter n.

Note

If HPMN is equal to [PMCR.N](#), this applies to all event counters.

If HPMN is less than [PMCR.N](#), for an event counter in the range [HPMN..[PMCR.N](#)-1]:

For an event counter <n> in the second range:

- The counter is accessible **only** from EL2 and **EL3 from Secure state**.
- If EL2 is disabled in the current Security state, the event counter is also accessible from EL1, and from EL0 if permitted by [PMUSERENR](#).
- If FEAT_PMUv3p5 is implemented, [HDCR.HLP](#) determines whether the counter overflows at [PMEVCNTR<n>\[31:0\]](#) or [PMEVCNTR<n>\[63:0\]](#).
- [HDCR.HPME](#) **and enables the operation of counters in this range**. [PMCNTENSET\[n\]](#) enable the operation of event counter n.

If [HPMN](#) **this field is set to 0, or to a value** larger than [PMCR.N](#), **or then if the following** [FEAT_HPMN0](#) is not implemented and HPMN is 0, the following CONSTRAINED UNPREDICTABLE behaviors apply:

- The value returned by a direct read of [HDCR.HPMN](#) is UNKNOWN.
- Either:
 - An UNKNOWN number of counters are reserved for EL2 use. That is, the PE behaves as if [HDCR.HPMN](#) is set to an UNKNOWN non-zero value less than or equal to [PMCR.N](#).
 - All counters are reserved for EL2 use, meaning no counters are accessible from Non-secure EL1 and Non-secure EL0.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to the value in [PMCR.N](#).

Otherwise:

Reserved, RES0.

Accessing HDCR

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	0b0001	0b0001	0b001

```

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);
    else
        UNDEFINED;
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, 0x03);
    else
        return HDCR;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        return HDCR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b100	0b0001	0b0001	0b001

```

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);
    else
        UNDEFINED;
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, 0x03);
    else
        HDCR = R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        HDCR = R[t];

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

HSCTLR, Hyp System Control Register

The HSCTLR characteristics are:

Purpose

Provides top level control of the system operation in Hyp mode.

Configuration

AArch32 System register HSCTLR bits [31:0] are architecturally mapped to AArch64 System register [SCTLR_EL2\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to HSCTLR are UNDEFINED.

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

Attributes

HSCTLR 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
DSSBS	TE	RES1	RES0	EE	RES0	RES1	RES0	WXN	RES1	RES0	RES1	RES0	I	RES1	RES0	SED	ITD	RES0	CP15	BEN	LSMA	OE	h	TL				

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 Hyp mode.
0b1	PSTATE.SSBS is set to 1 on an exception to Hyp mode.

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 EL2 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.

Bits [29:28]

Reserved, RES1.

Bits [27:26]

Reserved, RES0.

EE, bit [25]

The value of the PSTATE.E bit on entry to Hyp mode, the endianness of stage 1 translation table walks in the EL2 translation regime, and the endianness of stage 2 translation table walks in the PL1&0 translation regime.

The possible values of this bit are:

EE	Meaning
0b0	Little-endian. PSTATE.E is cleared to 0 on entry to Hyp mode. Stage 1 translation table walks in the EL2 translation regime, and stage 2 translation table walks in the PL1&0 translation regime are little-endian.
0b1	Big-endian. PSTATE.E is set to 1 on entry to Hyp mode. Stage 1 translation table walks in the EL2 translation regime, and stage 2 translation table walks in the PL1&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 reset behavior of this field is:

- On a Warm reset, this field resets to an IMPLEMENTATION DEFINED value.

Bit [24]

Reserved, RES0.

Bits [23:22]

Reserved, RES1.

Bits [21:20]

Reserved, RES0.

WXN, bit [19]

Write permission implies XN (Execute-never). For the EL2 translation regime, this bit can force all memory regions that are writable to be treated as XN. The possible values of this bit are:

WXN	Meaning
0b0	This control has no effect on memory access permissions.
0b1	Any region that is writable in the EL2 translation regime is forced to XN for accesses from software executing at EL2.

This bit applies only when HSCTLR.M bit is set.

ThisThe 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.

Bit [18]

Reserved, RES1.

Bit [17]

Reserved, RES0.

Bit [16]

Reserved, RES1.

Bits [15:13]

Reserved, RES0.

I, bit [12]

Instruction access Cacheability control, for accesses at EL2:

I	Meaning
0b0	All instruction access to Normal memory from EL2 are Non-cacheable for all levels of instruction and unified cache. If the value of HSTCLR.M is 0, instruction accesses from stage 1 of the EL2 translation regime are to Normal, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable memory.
0b1	All instruction access to Normal memory from EL2 can be cached at all levels of instruction and unified cache. If the value of HSTCLR.M is 0, instruction accesses from stage 1 of the EL2 translation regime are to Normal, Outer Shareable, Inner Write-Through, Outer Write-Through memory.

This bit has no effect on the PL1&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.

Bit [11]

Reserved, RES1.

Bits [10:9]

Reserved, RES0.

SED, bit [8]

SETEND instruction disable. Disables SETEND instructions at EL2.

SED	Meaning
0b0	SETEND instruction execution is enabled at EL2.
0b1	SETEND instructions are UNDEFINED at EL2.

If the implementation does not support mixed-endian operation at EL2, 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.

ITD, bit [7]

IT Disable. Disables some uses of IT instructions at EL2.

ITD	Meaning
0b0	All IT instruction functionality is enabled at EL2.
0b1	Any attempt at EL2 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'. 10100xxxxxxxxxxxx: ADD Rd, PC, #imm 01001xxxxxxxxxxxx: 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 HSCTLR then it must also be implemented in the [SCTLR_EL1](#), [SCTLR_EL2](#), 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**.

Bit [6]

Reserved, RES0.

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 EL2:

CP15BEN	Meaning
0b0	EL2 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is UNDEFINED.
0b1	EL2 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is enabled.

CP15BEN is optional, but if it is implemented in the HSCTLR then it must also be implemented in the [SCTLR_EL1](#), [SCTLR_EL2](#), 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 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 EL2, 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 EL2 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 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 EL2 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 EL2 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 1.

Otherwise:

Reserved, RES1.

C, bit [2]

Cacheability control, for data accesses at EL2:

C	Meaning
0b0	All data access to Normal memory from EL2, and all accesses to the EL2 translation tables, are Non-cacheable for all levels of data and unified cache.
0b1	All data access to Normal memory from EL2, and all accesses to the EL2 translation tables, can be cached at all levels of data and unified cache.

This bit has no effect on the PL1&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:

A	Meaning
0b0	Alignment fault checking disabled when executing at EL2. 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 or data elements being accessed.
0b1	Alignment fault checking enabled when executing at EL2. 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 or data elements 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, in a system where the PE resets into EL2, this field resets to an architecturally UNKNOWN value.

M, bit [0]

MMU enable for EL2 stage 1 address translation. Possible values of this bit are:

M	Meaning
0b0	EL2 stage 1 address translation disabled. See the HSCTLR.I field for the behavior of instruction accesses to Normal memory.
0b1	EL2 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 HSCTLR

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	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);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return HSCTLR;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        return HSCTLR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b100	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);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HSCTLR = R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        HSCTLR = R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd9b36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

HSR, Hyp Syndrome Register

The HSR characteristics are:

Purpose

Holds syndrome information for an exception taken to Hyp mode.

Configuration

AArch32 System register HSR bits [31:0] are architecturally mapped to AArch64 System register [ESR_EL2\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to HSR are UNDEFINED.

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

Attributes

HSR 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
EC						IL	ISS																								

Execution in any Non-secure PE mode other than Hyp mode makes this register UNKNOWN.

When an UNPREDICTABLE instruction is treated as UNDEFINED, and the exception is taken to EL2, the value of HSR is UNKNOWN. The value written to HSR 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.

EC, bits [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about. Possible values of this field are:

EC	Meaning	ISS
0b000000	Unknown reason.	ISS encoding for exceptions with an unknown reason
0b000001	Trapped WFI or WFE instruction execution. Conditional WFE and WFI instructions that fail their condition code check do not cause an exception.	ISS encoding for Exception from a WFI or WFE instruction
0b000011	Trapped MCR or MRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS encoding for Exception from an MCR or MRC access
0b000100	Trapped MCRR or MRRC access with (coproc==0b1111) that is not reported using EC 0b000000.	ISS encoding for Exception from an MCRR or MRRC access
0b000101	Trapped MCR or MRC access with (coproc==0b1110).	ISS encoding for Exception from an MCR or MRC access
0b000110	Trapped LDC or STC access. The only architected uses of these instructions 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 Exception from an LDC or STC instruction
0b000111	Access to Advanced SIMD or floating-point functionality trapped by a HCPTR .{TASE, TCP10} control. Excludes exceptions generated because Advanced SIMD and floating-point are not implemented. These are reported with EC value 0b000000.	ISS encoding for Exception from an access to SIMD or floating-point functionality, resulting from HCPTR
0b001000	Trapped VMRS access, from ID group trap, that is not reported using EC 0b000111.	ISS encoding for Exception from an MCR or MRC access
0b001100	Trapped MRRC access with (coproc==0b1110).	ISS encoding for Exception from an MCRR or MRRC access
0b001110	Illegal exception return to AArch32 state.	ISS encoding for Exception from an Illegal state or PC alignment fault
0b010001	Exception on SVC instruction execution in AArch32 state routed to EL2.	ISS encoding for Exception from HVC or SVC instruction execution
0b010010	HVC instruction execution in AArch32 state, when HVC is not disabled.	ISS encoding for Exception from HVC or SVC instruction execution
0b010011	Trapped execution of SMC instruction in AArch32 state.	ISS encoding for Exception from SMC instruction execution
0b100000	Prefetch Abort from a lower Exception level.	ISS encoding for Exception from a Prefetch Abort
0b100001	Prefetch Abort taken without a change in Exception level.	ISS encoding for Exception from a Prefetch Abort
0b100010	PC alignment fault exception.	ISS encoding for Exception from an

0b100100	Data Abort from a lower Exception level.	Illegal state or PC alignment fault
0b100101	Data Abort taken without a change in Exception level.	ISS encoding for Exception from a Data Abort

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 bit. Indicates the size of the instruction that has been trapped to Hyp mode. When this bit is valid, possible values of this bit are:

IL	Meaning
0b0	16-bit instruction trapped.
0b1	32-bit instruction trapped.

This field is RES1 and not valid for the following cases:

- When the EC field is 0b000000, indicating an exception with an unknown reason.
- Prefetch Aborts.
- Data Aborts for which the HSR.ISS.ISV field is 0.
- When the EC value is 0b001110, indicating an Illegal state exception.

Note

This is a change from the behavior in Armv7, where the IL field is UNK/SBZP for the corresponding cases.

The IL field is not valid and is UNKNOWN on an exception from a PC alignment fault.

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.

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.

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 is not accessible in the current PE mode in the current Security state, including:
 - A read access using a System register encoding pattern that is not allocated for reads or that does not permit reads in the current PE mode and Security state.
 - A write access using a System register encoding pattern that is not allocated for writes or that does not permit writes in the current PE mode and Security state.
 - Instruction encodings that are unallocated.
 - Instruction encodings for instructions 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.
- The 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](#).{ITD, SED, CP15BEN} control bits.
- Attempted execution of:
 - An HVC instruction when disabled by [HCR](#).HCD, [SCR](#).HCE, or [SCR_EL3](#).HCE.
 - An SMC instruction when disabled by [SCR](#).SCD or [SCR_EL3](#).SMD.
 - An HLT instruction when disabled by [EDSCR](#).HDE.
- An HVC instruction when disabled by [HCR](#).HCD, [SCR](#).HCE, or [SCR_EL3](#).HCE. An SMC instruction when disabled by [SCR](#).SCD or [SCR_EL3](#).SMD. An HLT instruction when disabled by [EDSCR](#).HDE.
- An exception generated because of the attempted execution of an MSR (Banked register) or MRS (Banked register) instruction that would access a Banked register that is not accessible from the Security state and PE mode at which the instruction was executed.

Note

An exception is generated only if the `CONSTRAINED UNPREDICTABLE` behavior of the instruction is that it is `UNDEFINED`, see 'MSR (banked register) and MRS (banked register)'.

- Attempted execution, in Debug state, of:
 - A DCPS1 instruction in Non-secure state from EL0 when EL2 is using AArch32 and the value of [HCR](#).TGE is 1.
 - A DCPS2 instruction at EL1 or EL0 when EL2 is not implemented, or when EL3 is using AArch32 and the value of [SCR](#).NS is 0, or when EL3 is using AArch64 and the value of [SCR_EL3](#).NS is 0.
 - A DCPS3 instruction when EL3 is not implemented, or when the value of [EDSCR](#).SDD is 1.
- 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.

'Undefined Instruction exception, when the value of [HCR](#).TGE is 1' describes the configuration settings for a trap that returns an [HSR](#).EC value of 0b000000.

ISS encoding for Exception from a WFI or WFE 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																			TI

CV, bit [24]

Condition code valid. Possible values of this bit are:

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

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. For more information, see the description of the COND field.

The reset behavior of this field is:

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

COND, bits [23:20]

The condition code for the trapped instruction.

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 [19:1]

Reserved, RES0.

TI, bit [0]

Trapped instruction. Possible values of this bit are:

TI	Meaning
0b0	WFI trapped.
0b1	WFE trapped.

The reset behavior of this field is:

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

'Traps to Hyp mode of Non-secure EL0 and EL1 execution of WFE and WFI instructions' describes the configuration settings for this trap.

ISS encoding for 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			RES0	Rt			CRm			Direction			

CV, bit [24]

Condition code valid. Possible values of this bit are:

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

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. For more information, see the description of the COND field.

The reset behavior of this field is:

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

COND, bits [23:20]

The condition code for the trapped instruction.

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.

Bit [9]

Reserved, RES0.

Rt, bits [8:5]

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.

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. **The possible values of this bit are:**

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 sections describe configuration settings for traps that are reported using EC value 0b000011:

- 'Traps to Hyp mode of Non-secure EL0 and EL1 accesses to the ID registers'.
- 'Traps to Hyp mode of Non-secure EL0 and EL1 accesses to lockdown, DMA, and TCM operations'.
- 'Traps to Hyp mode of Non-secure EL1 execution of cache maintenance instructions'.
- 'Traps to Hyp mode of Non-secure EL1 execution of TLB maintenance instructions'.
- 'Traps to Hyp mode of Non-secure EL1 accesses to the Auxiliary Control Register'.
- 'Traps to Hyp mode of Non-secure EL0 and EL1 accesses to Performance Monitors registers'.
- 'Traps to Hyp mode of Non-secure EL0 and EL1 accesses to Activity Monitors registers'.
- 'Traps to Hyp mode of Non-secure EL1 accesses to the CPACR'.
- 'Traps to Hyp mode of Non-secure EL1 accesses to virtual memory control registers'.
- 'General trapping to Hyp mode of Non-secure EL0 and EL1 accesses to System registers in the (coproc == 1111) encoding space'.

The following sections describe configuration settings for traps that are reported using EC value 0b000101:

- 'ID group 0, Primary device identification registers'.
- 'Traps to Hyp mode of Non-secure System register accesses to trace registers'.
- 'Trapping Non-secure System register accesses to Debug ROM registers'.
- 'Trapping Non-secure System register accesses to powerdown debug registers'.
- 'Trapping general Non-secure System register accesses to debug registers'.

The following sections describes configuration settings for traps that are reported using EC value 0b001000:

- 'ID group 0, Primary device identification registers'.
- 'ID group 3, Detailed feature identification registers'.

ISS encoding for 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				RES0	Rt				CRm				Direction	

CV, bit [24]

Condition code valid. Possible values of this bit are:

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

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. For more information, see the description of the COND field.

The reset behavior of this field is:

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

COND, bits [23:20]

The condition code for the trapped instruction.

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.

Bits [15:14]

Reserved, RES0.

Rt2, bits [13:10]

The Rt2 value from the issued instruction, the second 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.

Bit [9]

Reserved, RES0.

Rt, bits [8:5]

The Rt value from the issued instruction, the first 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. **The possible values of this bit are:**

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 sections describe configuration settings for traps that are reported using EC value 0b000100:

- 'Traps to Hyp mode of Non-secure EL1 accesses to virtual memory control registers'.
- 'Traps to Hyp mode of Non-secure EL0 and EL1 accesses to Performance Monitors registers'.
- 'Traps to Hyp mode of Non-secure EL0 and EL1 accesses to Activity Monitors registers'.
- 'Traps to Hyp mode of Non-secure EL0 and EL1 accesses to the Generic Timer registers'.
- 'General trapping to Hyp mode of Non-secure EL0 and EL1 accesses to System registers in the (coproc == 1111) encoding space'.

The following sections describe configuration settings for traps that are reported using EC value 0b001100:

- 'Traps to Hyp mode of Non-secure System register accesses to trace registers'.
- 'Trapping Non-secure System register accesses to Debug ROM registers'.

ISS encoding for 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. Possible values of this bit are:

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

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. For more information, see the description of the COND field.

The reset behavior of this field is:

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

COND, bits [23:20]

The condition code for the trapped instruction.

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 [11:9]

Reserved, RES0.

Rn, bits [8:5]

The Rn value from the issued instruction. 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	Literal unindexed. LDC instruction in A32 instruction set only. For a trapped STC instruction or a trapped T32 LDC instruction this encoding is reserved.
0b110	Literal offset. LDC instruction only. 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.

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. ~~The possible values of this bit are:~~

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.

'Trapping general Non-secure System register accesses to debug registers' describes the configuration settings for the trap that is reported using EC value 0b000110.

ISS encoding for Exception from an access to SIMD or floating-point functionality, resulting from HCPTR

24	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										TA RES0		coproc							

Excludes exceptions that occur because Advanced SIMD and floating-point functionality is not implemented, or because the value of [HCR.TGE](#) or [HCR_EL2.TGE](#) is 1. These are reported with EC value 0b000000.

CV, bit [24]

Condition code valid. Possible values of this bit are:

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

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. For more information, see the description of the COND field.

The reset behavior of this field is:

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

COND, bits [23:20]

The condition code for the trapped instruction.

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 [19:6]

Reserved, RES0.

TA, bit [5]

Indicates trapped use of Advanced SIMD functionality. ~~The possible values of this bit are:~~

TA	Meaning
0b0	Exception was not caused by trapped use of Advanced SIMD functionality.
0b1	Exception was caused by trapped use of Advanced SIMD functionality.

Any use of an Advanced SIMD instruction that is not also a floating-point instruction that is trapped to Hyp mode because of a trap configured in the [HCPTR](#) sets this bit to 1.

For a list of these instructions, see 'Controls of Advanced SIMD operation that do not apply to floating-point operation'.

The reset behavior of this field is:

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

Bit [4]

Reserved, RES0.

coproc, bits [3:0]

When the [HSR.TA](#) field returns the value 1, this field returns the value 0b1010. Otherwise, this field is RES0.

The reset behavior of this field is:

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

The following sections describe the configuration settings for the traps that are reported using EC value 0b000111:

- 'General trapping to Hyp mode of Non-secure accesses to the SIMD and floating-point registers'.
- 'Traps to Hyp mode of Non-secure accesses to Advanced SIMD functionality'.

ISS encoding for 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, this is the value of the imm16 field of the issued instruction.

For an 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.
- 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.

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.

'Supervisor Call exception, when the value of HCR.TGE is 1' describes the configuration settings for the trap reported with EC value 0b010001.

ISS encoding for Exception from SMC 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
CV	COND				CCKNOWNPASS	RES0																		

CV, bit [24]

Condition code valid. Possible values of this bit are:

CV	Meaning
0b0	The COND field is not valid.
0b1	The COND field is valid.

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. For more information, see the description of the COND field.

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]

The condition code for the trapped instruction.

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.

The reset behavior of this field is:

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

Bits [18:0]

Reserved, RES0.

'Traps to Hyp mode of Non-secure EL1 execution of SMC instructions' describes the configuration settings for this trap, for instructions executed in Non-secure EL1.

ISS encoding for Exception from a Prefetch 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														FnV	EA	RES0	S1PTW	RES0	IFSC							

Bits [24:11]

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	HIFAR is valid.
0b1	HIFAR 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. Possible values of this field are:

IFSC	Meaning	Applies when
0b000000	Address size fault in translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
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.	
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk.	
0b010101	Synchronous External abort on translation table walk, level 1.	
0b010110	Synchronous External abort on translation table walk, level 2.	
0b010111	Synchronous External abort on translation table walk, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk, level 3.	When FEAT_RAS is not implemented
0b100010	Debug exception.	
0b110000	TLB conflict abort.	

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults on a Long-descriptor translation table lookup'.

If the S1PTW bit is set, then the level refers the level of the stage2 translation that is translating a stage 1 translation walk.

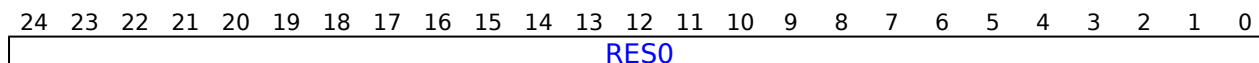
The reset behavior of this field is:

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

The following sections describe cases where Prefetch Abort exceptions can be routed to Hyp mode, generating exceptions that are reported in the HSR with EC value 0b100000:

- 'Abort exceptions, when the value of HCR.TGE is 1'.
- 'Routing debug exceptions to EL2 using AArch32'.

ISS encoding for Exception from an Illegal state or PC alignment fault



Bits [24:0]

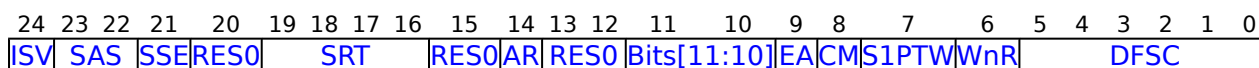
Reserved, RES0.

For more information about the Illegal state exception, see:

- 'Illegal changes to PSTATE.M'.
- 'Illegal return events from AArch32 state'.
- 'Legal returns that set PSTATE.IL to 1'.
- 'The Illegal Execution state exception'.

For more information about the PC alignment fault exception, see 'Branching to an unaligned PC'.

ISS encoding for Exception from a Data Abort



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.

This bit is 0 for all faults except Data Aborts generated by stage 2 address translations for which all the following apply to the instruction that generated the Data Abort exception:

- 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.
- The instruction is not performing register writeback.
- The instruction is not using the PC as a source or destination register.

For these cases, ISV is UNKNOWN if the exception was generated in Debug state in memory access mode, as described in 'Data Aborts in Memory access mode', and otherwise indicates whether ISS[23:14] hold a valid syndrome.

Note

In the A32 instruction set, LDR*T and STR*T instructions always perform register writeback and therefore never return a valid instruction syndrome.

When FEAT_RAS is implemented, ISV is 0 for any synchronous External abort.

ISV is set to 0 on a stage 2 abort on a stage 1 translation table walk.

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.

The reset behavior of this field is:

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

SAS, bits [23:22]

Syndrome Access Size. When ISV is 1, indicates the size of the access attempted by the faulting operation.

SAS	Meaning
0b00	Byte
0b01	Halfword
0b10	Word
0b11	Doubleword

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

The reset behavior of this field is:

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

SSE, bit [21]

Syndrome Sign Extend. When ISV is 1, for a byte, halfword, or word load operation, indicates whether the data item must be sign extended. For these cases, the possible values of this bit are:

SSE	Meaning
0b0	Sign-extension not required.
0b1	Data item must be sign-extended.

For all other operations this bit is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

The reset behavior of this field is:

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

Bit [20]

Reserved, RES0.

SRT, bits [19:16]

Syndrome Register transfer. When ISV is 1, the register number of the Rt operand of the faulting instruction.

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

The reset behavior of this field is:

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

Bit [15]

Reserved, RES0.

AR, bit [14]

Acquire/Release. When ISV is 1, the possible values of this bit are:

AR	Meaning
0b0	Instruction did not have acquire/release semantics.
0b1	Instruction did have acquire/release semantics.

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

The reset behavior of this field is:

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

Bits [13:12]

Reserved, RES0.

AET, bits [11:10]

When FEAT_RAS is implemented:

Asynchronous Error Type. When DFSC is 0b010001, describes the PE error state after taking the SError interrupt exception. **The possible values of this field are:**

AET	Meaning
0b00	Uncontainable (UC).
0b01	Unrecoverable state (UEU).
0b10	Restartable state (UEO).
0b11	Recoverable state (UER).

On a synchronous Data Abort, this field is RES0.

In the event of multiple errors 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.

When FEAT_RAS is not implemented, or when DFSC is not 0b010001:

- Bit[11] is RES0.
- Bit[10] forms the FnV field.

Note

Armv8.2 requires the implementation of FEAT_RAS.

The reset behavior of this field is:

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

FAR not Valid, for a synchronous External abort other than a synchronous External abort on a translation table walk.

FnV	Meaning
0b0	HDFAR is valid.
0b1	HDFAR is not valid, and holds an UNKNOWN value.

When FEAT_RAS is not implemented, this field is valid only if DFSC is 0b010000. It is RES0 for all other aborts.

When FEAT_RAS is implemented:

- If DFSC is 0b010000, this field is valid.
- If DFSC is 0b010001, this bit forms part of the AET field, becoming AET[0].
- This field is RES0 for all other aborts.

Note

Armv8.2 requires the implementation of FEAT_RAS.

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. For a synchronous fault, identifies fault that comes from a cache maintenance or address translation instruction. For synchronous faults, the possible values of this bit are:

CM	Meaning
0b0	Fault not generated by a cache maintenance or address translation instruction.
0b1	Fault generated by a cache maintenance or address translation instruction.

For an asynchronous Data Abort exception, this bit is 0.

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 a write instruction or a read instruction. ~~The possible values of this bit are:~~

WnR	Meaning
0b0	Abort caused by a read instruction.
0b1	Abort caused by a write instruction.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

On an asynchronous Data Abort:

- When FEAT_RAS is not implemented, this bit is UNKNOWN.
- When FEAT_RAS is implemented, this bit is RES0.

Note

Armv8.2 requires the implementation of FEAT_RAS.

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. Possible values of this field are:

DFSC	Meaning	Applies when
0b000000	Address size fault in translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
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.	
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010000	Synchronous External abort, not on translation table walk.	
0b010001	Asynchronous SError interrupt.	
0b010101	Synchronous External abort on translation table walk, level 1.	
0b010110	Synchronous External abort on translation table walk, level 2.	
0b010111	Synchronous External abort on translation table walk, level 3.	
0b011000	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b011001	Asynchronous SError interrupt, from a parity or ECC error on memory access.	When FEAT_RAS is not implemented
0b011101	Synchronous parity or ECC error on memory access on translation table walk, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk, level 3.	When FEAT_RAS is not implemented
0b100001	Alignment fault.	
0b100010	Debug exception.	
0b110000	TLB conflict abort.	
0b110100	IMPLEMENTATION DEFINED fault (Lockdown).	
0b110101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive access).	

All other values are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults on a Long-descriptor translation table lookup'.

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 describe cases where Data Abort exceptions can be routed to Hyp mode, generating exceptions that are reported in the HSR with EC value 0b100100:

- 'Abort exceptions, when the value of HCR.TGE is 1'.
- 'Routing debug exceptions to EL2 using AArch32'.

The following describe cases that can cause a Data Abort exception that is taken to Hyp mode, and reported in the HSR with EC value of 0b100000 or 0b100100:

- 'Hyp mode control of Non-secure access permissions'.
- 'Memory fault reporting in Hyp mode'.

Accessing HSR

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	0b0101	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T5 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T5 == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return HSR;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        return HSR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b100	0b0101	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T5 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T5 == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HSR = R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        HSR = R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbfdb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

HTCR, Hyp Translation Control Register

The HTCR characteristics are:

Purpose

The control register for stage 1 of the EL2 translation regime.

Note

This stage of translation always uses the Long-descriptor translation table format.

Configuration

AArch32 System register HTCR bits [31:0] are architecturally mapped to AArch64 System register [TCR_EL2\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to HTCR are UNDEFINED.

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

Attributes

HTCR 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
RES1	IMPLEMENTATION DEFINED	RES0	HWU62	HWU61	HWU60	HWU59	HPD	RES1	RES0				SH0	ORGN0	IRGN0	RES0	T0S													

Bit [31]

Reserved, RES1.

IMPLEMENTATION DEFINED, bit [30]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

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

Bit [29]

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 HTCR.HPD is 1.

The Effective value of this field is 0 if the value of HTCR.HPD is 0.

The reset behavior of this field is:

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

Otherwise:

Reserved, RAZ/WI.

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 HTCR.HPD is 1.

The Effective value of this field is 0 if the value of HTCR.HPD is 0.

The reset behavior of this field is:

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

Otherwise:

Reserved, RAZ/WI.

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 HTCR.HPD is 1.

The Effective value of this field is 0 if the value of HTCR.HPD is 0.

The reset behavior of this field is:

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

Otherwise:

Reserved, RAZ/WI.

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 HTCR.HPD is 1.

The Effective value of this field is 0 if the value of HTCR.HPD is 0.

The reset behavior of this field is:

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

Otherwise:

Reserved, RAZ/WI.

Reserved, RES0.

HPD, bit [24]

When FEAT_AA32HPD is implemented:

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, XNTable, and PXNTable, in the PL2 translation regime.

HPD	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.

Bit [23]

Reserved, RES1.

Bits [22:14]

Reserved, RES0.

SH0, bits [13:12]

Shareability attribute for memory associated with translation table walks using [HTTBR](#).

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 [HTTBR](#).

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 [HTTBR](#).

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 [7:3]

Reserved, RES0.

T0SZ, bits [2:0]

The size offset of the memory region addressed by [HTTBR](#). The region size is $2^{(32-T0SZ)}$ bytes.

The reset behavior of this field is:

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

Accessing HTCR

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	0b0010	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T2 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T2 == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return HTCR;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        return HTCR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b100	0b0010	0b0000	0b010

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T2 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T2 == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    HTCR = R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        UNDEFINED;
    else
        HTCR = R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICC_MSRE, Interrupt Controller Monitor System Register Enable register

The ICC_MSRE characteristics are:

Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL3.

Configuration

AArch32 System register ICC_MSRE bits [31:0] can be mapped to AArch64 System register [ICC_SRE_EL3\[31:0\]](#), but this is not architecturally mandated.

This register is present only when AArch32 is supported, FEAT_GICv3 is implemented and EL3 is implemented. Otherwise, direct accesses to ICC_MSRE are UNDEFINED.

Attributes

ICC_MSRE 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																		Enable		DIB	DFB	SRE									

Bits [31:4]

Reserved, RES0.

Enable, bit [3]

Enable. Enables lower Exception level access to [ICC_SRE](#) and [ICC_HSRE](#).

Enable	Meaning
0b0	Secure EL1 accesses to Secure ICC_SRE trap to EL3. EL2 accesses to Non-secure ICC_SRE and ICC_HSRE trap to EL3. Non-secure EL1 accesses to ICC_SRE trap to EL3, unless these accesses are trapped to EL2 as a result of ICC_HSRE.Enable == 0 .
0b1	Secure EL1 accesses to Secure ICC_SRE do not trap to EL3. EL2 accesses to Non-secure ICC_SRE and ICC_HSRE do not trap to EL3. Non-secure EL1 accesses to ICC_SRE do not trap to EL3.

If ICC_MSRE.SRE is RAO/WI, an implementation is permitted to make the Enable bit RAO/WI.

If ICC_MSRE.SRE is 0, the Enable bit behaves as 1 for all purposes other than reading the value of the bit.

The reset behavior of this field is:

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

DIB, bit [2]

Disable IRQ bypass.

DIB	Meaning
0b0	IRQ bypass enabled.
0b1	IRQ bypass disabled.

In systems that do not support IRQ bypass, this bit 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.

In systems that do not support FIQ bypass, this bit 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 EL3 or below to any ICH_* System register, or any EL1, EL2, or EL3 ICC_* register other than ICC_SRE , ICC_HSRE , or ICC_MSRE, are UNDEFINED.
0b1	The System register interface to the ICH_* registers and the EL1, EL2, and EL3 ICC_* registers is enabled for EL3.

If software changes this bit from 1 to 0, the results are UNPREDICTABLE.

If an implementation supports only a System register interface to the GIC CPU interface, this bit is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing ICC_MSRE

This register is always System register accessible.

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_MSRE.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.

This register is only accessible when executing in Monitor mode.

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	0b110	0b1100	0b1100	0b101
--------	-------	--------	--------	-------

```

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
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return ICC_MSRE;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b110	0b1100	0b1100	0b101

```

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
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' && CP15SDISABLE2 == HIGH then
        UNDEFINED;
    else
        ICC_MSRE = R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd36e47856c443a7ee9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ICIMVAU, Instruction Cache line Invalidate by VA to PoU

The ICIMVAU characteristics are:

Purpose

Invalidate instruction cache line by virtual address to PoU.

Configuration

AArch32 System instruction ICIMVAU performs the same function as AArch64 System instruction [IC IVAU](#).

This instruction is present only when AArch32 is supported. Otherwise, direct accesses to ICIMVAU are UNDEFINED.

Attributes

ICIMVAU 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
Virtual address to use																															

Bits [31:0]

Virtual address to use. No alignment restrictions apply to this VA.

Executing the ICIMVAU instruction

Execution of this instruction might require an address translation from VA to PA, and that translation might fault.

Execution of this instruction might require an address translation from VA to PA, and that translation might fault. For more information, see 'AArch32 instruction cache maintenance instructions (IC*)'.

If FEAT_CMOW is implemented, HCRX_EL2.CMOW is 1, and EL1 or EL0 access is enabled, when executed at EL1 or EL0, the instruction has stage 2 read permission to the VA but does not have stage 2 write permission to the VA, the instruction generates a stage 2 Permission fault.

For more information, see 'AArch32 instruction cache maintenance instructions (IC*)'.

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	0b0101	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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.TPU == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TOCU == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR.TPU == '1' then
        AArch32.TakeHypTrapException(0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HCR2.TOCU == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        AArch32.IC(R[t], CacheOpScope_PoU);
elsif PSTATE.EL == EL2 then
    AArch32.IC(R[t], CacheOpScope_PoU);
elsif PSTATE.EL == EL3 then
    AArch32.IC(R[t], CacheOpScope_PoU);

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 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 AArch32 is supported. 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 also includes support for: <ul style="list-style-type: none"> Extended 16-bit PMEVTYPER<n>.evtCount field. If EL2 is implemented, the HDCR.HPMD control control bit.
0b0101	PMUv3 for Armv8.4. As 0b0100, and adds also includes support for the PMMIR register.
0b0110	PMUv3 for Armv8.5. As 0b0101, and adds also includes support for: <ul style="list-style-type: none"> 64-bit event counters. If EL2 is implemented, the HDCR.HCCD control control bit. If EL3 is implemented, the SDCR.SCCD control control bit.
0b0111	PMUv3 for Armv8.7. As 0b0110, and adds also includes support for: <ul style="list-style-type: none"> The PMCR.FZO and, if EL2 is implemented, HDCR.HPMFZO controls control bits. If EL3 is implemented and using AArch64, the MDCR_EL3.{MPMX,MCCD} controls control bits.
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.
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.

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

In Armv7, the value 0b0000 can mean that PMUv1 is implemented. PMUv1 is 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:

Support for System registers-based debug model, using registers in the coproc == 0b1110 encoding space, for A and R-profile processors. Defined values are:

CopDBG	Meaning
0b0000	Not supported.
0b0010	Support for Armv6, v6 Debug architecture, with System registers access.
0b0011	Support for Armv6, v6.1 Debug architecture, with System registers access.
0b0100	Support for Armv7, v7 Debug architecture, with System registers access.
0b0101	Support for Armv7, v7.1 Debug architecture, with System registers access.
0b0110	Support for Armv8 debug architecture, with System registers access.
0b0111	Support for Armv8 debug architecture, with System registers access, and Virtualization Host Extensions.
0b1000	Support for Armv8.2 debug architecture, FEAT_Debugv8p2.
0b1001	Support for Armv8.4 debug architecture, FEAT_Debugv8p4.
0b1010	Armv8.8 debug architecture, FEAT_Debugv8p8.

All other values are reserved.

The values 0b0000, 0b0010, 0b0011, 0b0100, and 0b0101 are not permitted in Armv8.

FEAT_VHEFEAT_Debugv8p2 adds the functionality identified by the value 0b01110b1000.

FEAT_Debugv8p2FEAT_Debugv8p4 adds the functionality identified by the value 0b10000b1001.

FEAT_Debugv8p4 addsIn Armv8.0, the functionalityonly identifiedpermitted byvalue the valueis 0b10010b0110.

FEAT_Debugv8p8 addsIn Armv8.1, the functionalityonly identifiedpermitted byvalue the valueis 0b10100b0111.

FromIn Armv8.1Armv8.2, whenthe only permitted value is FEAT_VHE0b1000 is implemented the value 0b0110 is not permitted.;

From Armv8.2Armv8.4, the valuesonly permitted value is 0b01100b1001 and 0b0111 are not permitted.;

From Armv8.4, the value 0b1000 is not permitted.

From Armv8.8, the value 0b1001 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
        return ID_DFR0;
elsif PSTATE.EL == EL2 then
    return ID_DFR0;
elsif PSTATE.EL == EL3 then
    return ID_DFR0;

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ID_DFR1, Debug Feature Register 1

The ID_DFR1 characteristics are:

Purpose

Provides top level information about the debug system in AArch32.

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_DFR1 bits [31:0] are architecturally mapped to AArch64 System register [ID_DFR1_EL1\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to ID_DFR1 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_DFR1 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														HPMN0MTPMU				MTPMU			

Bits [31:8]

Reserved, RES0.

HPMN0, bits [7:4]

Zero PMU event counters for a Guest operating system. Defined values are:

HPMN0	Meaning
0b0000	Setting HDCR .HPMN to zero has CONSTRAINED UNPREDICTABLE behavior.
0b0001	Setting HDCR .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.

MTPMU, bits [3:0]

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>. are read/write or RES0.
0b0001	FEAT_MTPMU and FEAT_PMUv3 implemented. PMEVTYPER<n>. are read/write. When FEAT_MTPMU is disabled, the Effective values of PMEVTYPER<n>. are 0.
0b1111	FEAT_MTPMU not implemented. If FEAT_PMUv3 is implemented, PMEVTYPER<n>. 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.

Accessing ID_DFR1

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	0b0011	0b101

```

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) && (!IsZero(ID_DFR1) || boolean
IMPLEMENTATION_DEFINED "ID_DFR1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && (!IsZero(ID_DFR1) || boolean
IMPLEMENTATION_DEFINED "ID_DFR1 trapped by HCR.TID3") && HCR.TID3 == '1' then
        AArch32.TakeHypTrapException(0x03);
    else
        return ID_DFR1;
elsif PSTATE.EL == EL2 then
    return ID_DFR1;
elsif PSTATE.EL == EL3 then
    return ID_DFR1;

```

3020/09/2021 14:12:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbf36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

ID_PFR2, Processor Feature Register 2

The ID_PFR2 characteristics are:

Purpose

Gives information about the AArch32 programmers' model.

Must be interpreted with [ID_PFR0](#) and [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_PFR2 bits [31:0] are architecturally mapped to AArch64 System register [ID_PFR2_EL1\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to ID_PFR2 are UNDEFINED.

Attributes

ID_PFR2 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												RAS_frac				SSBS				CSV3											

Bits [31:12]

Reserved, RES0.

RAS_frac, bits [11:8]

RAS Extension fractional field.

RAS_frac	Meaning
0b0000	If ID_PFR0 .RAS == 0b0001, RAS Extension implemented.
0b0001	If ID_PFR0 .RAS == 0b0001, as 0b0000 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.

All other values are reserved.

This field is valid only if [ID_PFR0](#).RAS == 0b0001.

SSBS, bits [7:4]

Speculative Store Bypassing controls in AArch64 state. Defined values are:

SSBS	Meaning
0b0000	AArch32 provides no mechanism to control the use of Speculative Store Bypassing.
0b0001	AArch32 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypass Safe.

In Armv8.0, the permitted values are 0b0000 and 0b0001.

From Armv8.5, the only permitted value is 0b0001.

All other values are reserved.

CSV3, bits [3:0]

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, or 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.

Accessing ID_PFR2

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	0b0011	0b100

```

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
        return ID_PFR2;
elsif PSTATE.EL == EL2 then
    return ID_PFR2;
elsif PSTATE.EL == EL3 then
    return ID_PFR2;

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MPIDR, Multiprocessor Affinity Register

The MPIDR characteristics are:

Purpose

In a multiprocessor system, provides an additional PE identification mechanism for scheduling purposes.

Configuration

AArch32 System register MPIDR bits [31:0] are architecturally mapped to AArch64 System register [MPIDR_EL1\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to MPIDR are UNDEFINED.

In a uniprocessor system, Arm recommends that each Aff<n> field of this register returns a value of 0.

Attributes

MPIDR 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
M	U						RES0				MT																				

M, bit [31]

Indicates whether this implementation includes the functionality introduced by the [Arm v7-ARMv7](#) Multiprocessing Extensions.

M	Meaning
0b0	This implementation does not include the Arm v7-ARMv7 Multiprocessing Extensions functionality.
0b1	This implementation includes the Arm v7-ARMv7 Multiprocessing Extensions functionality.

[AccessFrom to Armv8](#), this [fieldbit](#) is [RAO](#). [RAO/WI](#).

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

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	0b0000	0b101

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T0 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T0 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif EL2Enabled() && !ELUsingAArch32(EL2) then
        return VMPIDR_EL2<31:0>;
    elseif EL2Enabled() && ELUsingAArch32(EL2) then
        return VMPIDR;
    else
        return MPIDR;
elseif PSTATE.EL == EL2 then
    return MPIDR;
elseif PSTATE.EL == EL3 then
    return MPIDR;

```

3020/09/2021 14:52:36; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MVBAR, Monitor Vector Base Address Register

The MVBAR characteristics are:

Purpose

When EL3 is implemented and can use AArch32, holds the vector base address for any exception that is taken to Monitor mode.

Secure software must program the MVBAR with the required initial value as part of the PE boot sequence.

Configuration

This register is present only when AArch32 is supported. Otherwise, direct accesses to MVBAR are UNDEFINED.

It is IMPLEMENTATION DEFINED whether MVBAR[0] has a fixed value and ignored writes, or takes the last value written to it.

On a Warm reset into EL3 using AArch32, the reset value of MVBAR is an IMPLEMENTATION DEFINED choice between the following:

- MVBAR[31:5] = an IMPLEMENTATION DEFINED value, which might be UNKNOWN, MVBAR[4:1] = RES0, and MVBAR[0] = 0.
- MVBAR[31:1] = an IMPLEMENTATION DEFINED value that is bits[31:1] of the AArch32 reset address, and MVBAR[0] = 1.

Attributes

MVBAR is a 32-bit register.

Field descriptions

When programmed with a vector base 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
Vector Base Address																												Reserved			

Bits [31:5]

Vector Base Address. Bits[31:5] of the base address of the exception vectors for exceptions taken to this Exception level. Bits[4:0] of an exception vector are the exception offset.

Reserved, bits [4:0]

Reserved, see Configurations.

Accessing MVBAR

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	0b0000	0b001
--------	-------	--------	--------	-------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if IsHighestEL(EL1) then
        return RVBAR;
    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 !ELUsingAArch32(EL2) && SCR_EL3.<NS,EEL2> == '01' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif !ELUsingAArch32(EL3) && SCR_EL3.NS == '0' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if IsHighestEL(EL2) then
        return RVBAR;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    return MVBAR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b0000	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);
    elsif !ELUsingAArch32(EL2) && SCR_EL3.<NS,EEL2> == '01' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif !ELUsingAArch32(EL3) && SCR_EL3.NS == '0' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' && CP15SDISABLE == HIGH then
        UNDEFINED;
    elsif SCR.NS == '0' && CP15SDISABLE2 == HIGH then
        UNDEFINED;
    else
        MVBAR = R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbbdfb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

NMRR, Normal Memory Remap Register

The NMRR characteristics are:

Purpose

Provides additional mapping controls for memory regions that are mapped as Normal memory by their entry in the [PRRR](#).

Used in conjunction with the [PRRR](#).

Configuration

AArch32 System register NMRR bits [31:0] are architecturally mapped to AArch64 System register [MAIR_EL1\[63:32\]](#) when EL3 is not implemented or EL3 is using AArch64.

AArch32 System register NMRR bits [31:0] are architecturally mapped to AArch32 System register [MAIR1\[31:0\]](#) when EL3 is not implemented or EL3 is using AArch64.

AArch32 System register NMRR bits [31:0] (NMRR_S) are architecturally mapped to AArch32 System register [MAIR1\[31:0\]](#) (MAIR1_S) when EL3 is using AArch32.

AArch32 System register NMRR bits [31:0] (NMRR_NS) are architecturally mapped to AArch32 System register [MAIR1\[31:0\]](#) (MAIR1_NS) when EL3 is using AArch32.

This register is present only when AArch32 is supported. Otherwise, direct accesses to NMRR are UNDEFINED.

[MAIR1](#) and NMRR are the same register, with a different view depending on the value of [TTBCR.EAE](#):

- When it is set to 0, the register is as described in NMRR.
- When it is set to 1, the register is as described in [MAIR1](#).

Attributes

NMRR 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
OR7	OR6	OR5	OR4	OR3	OR2	OR1	OR0	IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0																

OR<n>, bits [2n+17:2n+16], for n = 7 to 0

Outer Cacheable property mapping for memory attributes n, if the region is mapped as Normal memory by the PRRR.TR<n> entry. n is the value of the TEX[0], C, and B bits concatenated. ~~The possible values of this field are:~~

OR<n>	Meaning
0b00	Region is Non-cacheable.
0b01	Region is Write-Back, Write-Allocate.
0b10	Region is Write-Through, no Write-Allocate.
0b11	Region is Write-Back, no Write-Allocate.

The meaning of the field with n = 6 is IMPLEMENTATION DEFINED and might differ from the meaning given here. This is because the meaning of the attribute combination {TEX[0] = 1, C = 1, B = 0} is IMPLEMENTATION DEFINED.

When FEAT_XS is implemented, stage 1 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.

IR<n>, bits [2n+1:2n], for n = 7 to 0

Inner Cacheable property mapping for memory attributes n, if the region is mapped as Normal memory by the PRRR.TR<n> entry. n is the value of the TEX[0], C, and B bits concatenated. **The possible values of this field are:**

IR<n>	Meaning
0b00	Region is Non-cacheable.
0b01	Region is Write-Back, Write-Allocate.
0b10	Region is Write-Through, no Write-Allocate.
0b11	Region is Write-Back, no Write-Allocate.

The meaning of the field with n = 6 is IMPLEMENTATION DEFINED and might differ from the meaning given here. This is because the meaning of the attribute combination {TEX[0] = 1, C = 1, B = 0} is IMPLEMENTATION DEFINED.

When FEAT_XS is implemented, stage 1 Inner 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 NMRR

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	0b1010	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T10 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T10 == '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
        if TTBCR.EAE == '1' then
            return MAIR1_NS;
        else
            return NMRR_NS;
    else
        if TTBCR.EAE == '1' then
            return MAIR1;
        else
            return NMRR;
    elsif PSTATE.EL == EL2 then
        if HaveEL(EL3) && ELUsingAArch32(EL3) then
            if TTBCR.EAE == '1' then
                return MAIR1_NS;
            else
                return NMRR_NS;
        else
            if TTBCR.EAE == '1' then
                return MAIR1;
            else
                return NMRR;
    elsif PSTATE.EL == EL3 then
        if TTBCR.EAE == '1' then
            if SCR.NS == '0' then
                return MAIR1_S;
            else
                return MAIR1_NS;
        else
            if SCR.NS == '0' then
                return NMRR_S;
            else
                return NMRR_NS;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1010	0b0010	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T10 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T10 == '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
        if TTBCR.EAE == '1' then
            MAIR1_NS = R[t];
        else
            NMRR_NS = R[t];
        else
            if TTBCR.EAE == '1' then
                MAIR1 = R[t];
            else
                NMRR = R[t];
    elsif PSTATE.EL == EL2 then
        if HaveEL(EL3) && ELUsingAArch32(EL3) then
            if TTBCR.EAE == '1' then
                MAIR1_NS = R[t];
            else
                NMRR_NS = R[t];
        else
            if TTBCR.EAE == '1' then
                MAIR1 = R[t];
            else
                NMRR = 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 TTBCR.EAE == '1' then
                if SCR.NS == '0' then
                    MAIR1_S = R[t];
                else
                    MAIR1_NS = R[t];
            else
                if SCR.NS == '0' then
                    NMRR_S = R[t];
                else
                    NMRR_NS = R[t];

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

NSACR, Non-Secure Access Control Register

The NSACR characteristics are:

Purpose

When EL3 is implemented and can use AArch32, defines the Non-secure access permissions to Trace, Advanced SIMD and floating-point functionality. Also includes IMPLEMENTATION DEFINED bits that can define Non-secure access permissions for IMPLEMENTATION DEFINED functionality.

Configuration

This register is present only when AArch32 is supported. Otherwise, direct accesses to NSACR are UNDEFINED.

Note

In AArch64 state, the NSACR controls are replaced by controls in [CPTR_EL3](#).

Attributes

NSACR 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											NSTRCDIS	RES0	IMPLEMENTATION DEFINED		NSASEDIS	RES0	cp11	cp10	RES0												

If EL3 is implemented and is using AArch64 then:

- Any read of the NSACR from Non-secure EL2 or Non-secure EL1 returns a value of 0x00000C00.
- Any read or write to NSACR from Secure EL1 is trapped as an exception to EL3.

If EL3 is not implemented, then any read of the NSACR from EL2 or EL1 returns a value of 0x00000C00.

Bits [31:21]

Reserved, RES0.

NSTRCDIS, bit [20]

Disables Non-secure System register accesses to all implemented trace registers.

NSTRCDIS	Meaning
0b0	This control has no effect on: <ul style="list-style-type: none"> System register access to implemented trace registers. The behavior of CPACR.TRCDIS and HCPTR.TTA.
0b1	Non-secure System register accesses to all implemented trace registers are disabled, meaning: <ul style="list-style-type: none"> CPACR.TRCDIS behaves as RAO/WI in Non-secure state, regardless of its actual value. HCPTR.TTA behaves as RAO/WI, regardless of its actual value.

The implementation of this field must correspond to the implementation of the [CPACR](#).TRCDIS field:

- If [CPACR](#).TRCDIS is RAZ/WI, this field is RAZ/WI.
- If [CPACR](#).TRCDIS is RW, this field is RW.

Note

- The ETMv4 architecture and ETE do not permit EL0 to access the trace registers. If the PE trace unit implements FEAT_ETMv4 or FEAT_ETE, EL0 accesses to the trace registers are UNDEFINED.
- The Arm architecture does not provide Non-secure access controls on trace register accesses through the optional memory-mapped external debug 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.

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.

Bit [19]

Reserved, RES0.

IMPLEMENTATION DEFINED, bits [18:16]

IMPLEMENTATION DEFINED.

NSASEDIS, bit [15]

Disables Non-secure access to the Advanced SIMD functionality.

NSASEDIS	Meaning
0b0	This control has no effect on: <ul style="list-style-type: none"> • Non-secure access to Advanced SIMD functionality. • The behavior of CPACR.ASEDIS and HCPTR.TASE.
0b1	Non-secure access to the Advanced SIMD functionality is disabled, meaning: <ul style="list-style-type: none"> • CPACR.ASEDIS behaves as RAO/WI in Non-secure state, regardless of its actual value. • HCPTR.TASE behaves as RAO/WI, regardless of its actual value.

The implementation of this field must correspond to the implementation of the [CPACR](#).ASEDIS field:

- If [CPACR](#).ASEDIS is RES0, this field is RES0. If the implementation does not include Advanced SIMD and floating-point functionality, this field is RES0.
- If [CPACR](#).ASEDIS is RAZ/WI, this field is RAZ/WI.
- If [CPACR](#).ASEDIS is RW, this field is RW.

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.

Bits [14:12]

Reserved, RES0.

cp11, bit [11]

The value of this field is ignored. If this field is programmed with a different value to the cp10 field then this field is UNKNOWN on a direct read of the NSACR.

If the implementation does not include Advanced SIMD and floating-point functionality, this field is RES0.

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.

cp10, bit [10]

Enable Non-secure access to the Advanced SIMD and floating-point features. Possible values of the fields are:

cp10	Meaning
0b0	Advanced SIMD and floating-point features can be accessed only from Secure state. Any attempt to access this functionality from Non-secure state is UNDEFINED. When the PE is in Non-secure state: <ul style="list-style-type: none"> The CPACR.{cp11, cp10} fields ignore writes and read as 0b00, access denied. The HCPTR.{TCP11, TCP10} fields behave as RAO/WI, regardless of their actual values.
0b1	Advanced SIMD and floating-point features can be accessed from both Security states.

If Non-secure access to the Advanced SIMD and floating-point functionality is enabled, the [CPACR](#) must be checked to determine the level of access that is permitted.

The Advanced SIMD and floating-point features controlled by these fields are:

- Execution of any floating-point or Advanced SIMD instruction.
- Any access to the Advanced SIMD and floating-point registers D0-D31 and their views as S0-S31 and Q0-Q15.
- Any access to the [FPSCR](#), [FPSID](#), [MVER0](#), [MVER1](#), [MVER2](#), or [FPEXC](#) System registers.

If the implementation does not include Advanced SIMD and floating-point functionality, this field is RES0.

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.

Bits [9:0]

Reserved, RES0.

Accessing NSACR

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	0b0001	0b010

```

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 !ELUsingAArch32(EL2) && SCR_EL3.<NS,EEL2> == '01' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif !ELUsingAArch32(EL3) && SCR_EL3.NS == '0' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    elsif !HaveEL(EL3) || (!ELUsingAArch32(EL3) && SCR_EL3.NS == '1') then
        return Zeros(20):'1100':Zeros(8);
    else
        return NSACR;
elsif PSTATE.EL == EL2 then
    if !HaveEL(EL3) || (!ELUsingAArch32(EL3) && SCR_EL3.NS == '1') then
        return Zeros(20):'1100':Zeros(8);
    else
        return NSACR;
elsif PSTATE.EL == EL3 then
    return NSACR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0001	0b0001	0b010

```

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 !ELUsingAArch32(EL2) && SCR_EL3.<NS,EEL2> == '01' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif !ELUsingAArch32(EL3) && SCR_EL3.NS == '0' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' && CP15SDISABLE2 == HIGH then
        UNDEFINED;
    else
        NSACR = R[t];

```

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd36e47856e443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PAR, Physical Address Register

The PAR 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

AArch32 System register PAR bits [63:0] are architecturally mapped to AArch64 System register [PAR_EL1\[63:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to PAR are UNDEFINED.

PAR 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].

The Configurations section specifies the cases where each PAR format is used.

PAR is accessed as a 32-bit value:

- When the PE is not in Hyp mode and is using the Short-descriptor translation table format.
- When the PE is in Hyp mode and executes an [ATS12NSOPR](#), [ATS12NSOPW](#), [ATS12NSOUR](#), or [ATS12NSOUW](#) instruction and the value of [HCR.VM](#) is 0 and the value of [TTBCR.EAE](#) is 0.

In these cases, PAR[63:32] is RES0.

Otherwise, the PAR is accessed as a 64-bit value, if any of the following is true:

- When using the Long-descriptor translation table format.
- If the stage 1 address translation is disabled and [TTBCR.EAE](#) is set to 1.
- In an implementation that includes EL2, for the result of an ATS1Cxx instruction performed from Hyp mode.

For PL1&0 stage 1 translations, [TTBCR.EAE](#) selects the translation table format.

Attributes

PAR is a 64-bit register.

Field descriptions

When the instruction returned a 32-bit value to the PAR, PAR.F==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																																							
PA																				LPAEN		NOS		NS		IMPLEMENTATION DEFINED						SH	Inner[2:0]			Outer[1:0]		SS	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 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 can return a value that indicates the resulting attributes, rather than the values that appear in the translation table descriptors. More precisely:

- Memory attribute 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. This applies to the NOS, SH, Inner, and Outer fields.
- See the NS bit description for constraints on the value it returns.

Bits [63:32]

Reserved, RES0.

PA, bits [31:12]

Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[31:12].

The reset behavior of this field is:

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

LPAE, bit [11]

When updating the PAR with the result of the translation operation, this bit is set as follows:

LPAE	Meaning
0b0	Short-descriptor translation table format used. This means the PAR returned a 32-bit value.

The reset behavior of this field is:

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

NOS, bit [10]

Not Outer Shareable. When the returned value of PAR.SH is 1, indicates the Shareability attribute for the physical memory region:

NOS	Meaning
0b0	Memory region is Outer Shareable.
0b1	Memory region is Inner Shareable.

When the returned value of PAR.SH is 0 the value returned to this field is UNKNOWN.

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.

NS, bit [9]

Non-secure. The NS attribute for a translation table entry from a Secure translation regime.

For a result from a Secure translation regime, 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.

IMPLEMENTATION DEFINED, bit [8]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

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

SH, bit [7]

Shareability. Indicates whether the physical memory region is Non-shareable:

SH	Meaning
0b0	Memory is Non-shareable.
0b1	Memory is shareable, and PAR.NOS indicates whether the region is Outer Shareable or Inner Shareable.

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.

Inner[2:0], bits [6:4]

Inner cacheability attribute for the region. Permitted values are:

Inner[2:0]	Meaning
0b000	Non-cacheable.
0b001	Device-nGnRnE.
0b011	Device-nGnRE.
0b101	Write-Back, Write-Allocate.
0b110	Write-Through.
0b111	Write-Back, no Write-Allocate.

The values 0b010 and 0b100 are 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.

The reset behavior of this field is:

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

Outer[1:0], bits [3:2]

Outer cacheability attribute for the region. Permitted values are:

Outer[1:0]	Meaning
0b00	Non-cacheable.
0b01	Write-Back, Write-Allocate.
0b10	Write-Through, no Write-Allocate.
0b11	Write-Back, no Write-Allocate.

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.

SS, bit [1]

Supersection. Used to indicate if the result is a Supersection:

SS	Meaning
0b0	Result is not a Supersection. PAR[31:12] contains OA[31:12].
0b1	Result is a Supersection, and: <ul style="list-style-type: none"> PAR[31:24] contains OA[31:24]. PAR[23:16] contains OA[39:32]. PAR[15:12] contains 0b0000. If an implementation supports less than 40 bits of physical address, the bits in the PAR field that correspond to physical address bits that are not implemented are UNKNOWN.

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
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 the instruction returned a 32-bit value to the PAR, PAR.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	38	37	36	35	34	33	32	
																RES0																
IMPLEMENTATION DEFINED																RES0				LPAE		RES0				FS[5]		FS[4:0]				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 [63:32]

Reserved, RES0.

IMPLEMENTATION DEFINED, bits [31:16]

IMPLEMENTATION DEFINED.

The reset behavior of this field is:

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

Bits [15:12]

Reserved, RES0.

LPAE, bit [11]

When updating the PAR with the result of the translation operation, this bit is set as follows:

LPAE	Meaning
0b0	Short-descriptor translation table format used. This means the PAR returned a 32-bit value.

The reset behavior of this field is:

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

Bits [10:7]

Reserved, RES0.

FS[5], bit [6]

Fault status bits, External abort type. Provides an IMPLEMENTATION DEFINED classification of an External abort. Values are as in [in](#) the [DFSR](#).ExT field when using the Short-descriptor translation table format.

In an implementation that does not provide any classification of External aborts, this bit is RES0.

For aborts other than External aborts this bit always returns 0.

The reset behavior of this field is:

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

FS[4:0], bits [5:1]

Fault status bits. Values are as in the [DFSR](#).FS field when using the Short-descriptor translation table format.

FS[4:0]	Meaning	Applies when
0b00001	Alignment fault.	
0b00011	Access flag fault, level 1.	
0b00100	Fault on instruction cache maintenance.	
0b00101	Translation fault, level 1.	
0b00110	Access flag fault, level 2.	
0b00111	Translation fault, level 2.	
0b01001	Domain fault, level 1.	
0b01011	Domain fault, level 2.	
0b01100	Synchronous External abort, on translation table walk, level 1.	
0b01101	Permission fault, level 1.	
0b01110	Synchronous External abort, on translation table walk, level 2.	
0b01111	Permission fault, level 2.	
0b10000	TLB conflict abort.	
0b11001	Synchronous parity or ECC error on memory access, not on translation table walk.	When FEAT_RAS is not implemented
0b11100	Synchronous parity or ECC error on translation table walk, level 1.	When FEAT_RAS is not implemented
0b11110	Synchronous parity or ECC error on translation table walk, level 2.	When FEAT_RAS is not implemented

The reset behavior of this field is:

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

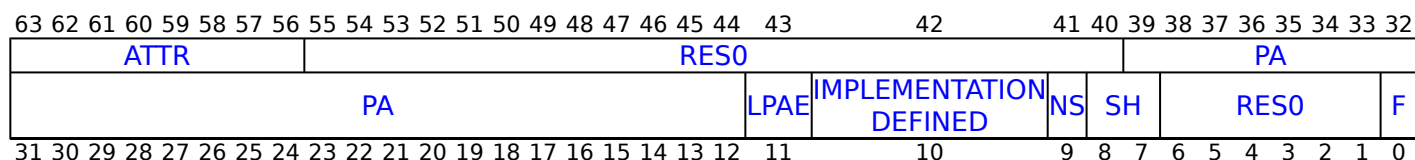
F, bit [0]

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 the instruction returned a 64-bit value to the PAR, PAR.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 can return a value that indicates the resulting attributes, rather than the values that appear in the translation table descriptors. More precisely:

- Memory attribute 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. This applies to the ATTR and SH fields.
- See the 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 [MAIR0](#) and [MAIR1](#).

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 [55:40]

Reserved, RES0.

PA, bits [39:12]

Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[39:12].

The reset behavior of this field is:

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

LPAE, bit [11]

When updating the PAR with the result of the translation operation, this bit is set as follows:

LPAE	Meaning
0b1	Long-descriptor translation table format used. This means the PAR returned a 64-bit value.

The reset behavior of this field is:

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

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]

Non-secure. The NS attribute for a translation table entry from a Secure translation regime.

For a result from a Secure translation regime, 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. Permitted values are:

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 the instruction returned a 64-bit value to the PAR, PAR.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	38	37	36	35	34	33	32								
IMPLEMENTATION DEFINED								IMPLEMENTATION DEFINED								IMPLEMENTATION DEFINED								RES0															
RES0																LPAE		RES0	FSTAGE	S2WLK	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.

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 [47:12]

Reserved, RES0.

LPAE, bit [11]

When updating the PAR with the result of the translation operation, this bit is set as follows:

LPAE	Meaning
0b1	Long-descriptor translation table format used. This means the PAR returned a 64-bit value.

The reset behavior of this field is:

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

Bit [10]

Reserved, RES0.

FSTAGE, bit [9]

Indicates the translation stage at which the translation aborted:

FSTAGE	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.

S2WLK, 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 field. Values are as in the [DFSR.STATUS](#) and [IFSR.STATUS](#) fields when using the Long-descriptor translation table format.

FST	Meaning	Applies when
0b000000	Address size fault in translation table base register.	
0b000001	Address size fault, level 1.	
0b000010	Address size fault, level 2.	
0b000011	Address size fault, level 3.	
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.	
0b001101	Permission fault, level 1.	
0b001110	Permission fault, level 2.	
0b001111	Permission fault, level 3.	
0b010101	Synchronous External abort on translation table walk, level 1.	
0b010110	Synchronous External abort on translation table walk, level 2.	
0b010111	Synchronous External abort on translation table walk, level 3.	
0b011101	Synchronous parity or ECC error on memory access on translation table walk, level 1.	When FEAT_RAS is not implemented
0b011110	Synchronous parity or ECC error on memory access on translation table walk, level 2.	When FEAT_RAS is not implemented
0b011111	Synchronous parity or ECC error on memory access on translation table walk, level 3.	When FEAT_RAS is not implemented
0b110000	TLB conflict abort.	

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

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	0b0111	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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 HaveEL(EL3) && ELUsingAArch32(EL3) then
        return PAR_NS<31:0>;
    else
        return PAR<31:0>;
elsif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        return PAR_NS<31:0>;
    else
        return PAR<31:0>;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        return PAR_S<31:0>;
    else
        return PAR_NS<31:0>;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0111	0b0100	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
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 HaveEL(EL3) && ELUsingAArch32(EL3) then
        PAR_NS = ZeroExtend(R[t]);
    else
        PAR = ZeroExtend(R[t]);
elsif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        PAR_NS = ZeroExtend(R[t]);
    else
        PAR = ZeroExtend(R[t]);
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        PAR_S = ZeroExtend(R[t]);
    else
        PAR_NS = ZeroExtend(R[t]);

```

MRRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm>

coproc	CRm	opc1
--------	-----	------

0b1111	0b0111	0b0000
--------	--------	--------

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T7 == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
        return PAR_NS;
    else
        return PAR;
elsif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        return PAR_NS;
    else
        return PAR;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        return PAR_S;
    else
        return PAR_NS;

```

MCCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <Rt2>, <CRm>

coproc	CRm	opc1
0b1111	0b0111	0b0000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T7 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x04);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T7 == '1' then
        AArch32.TakeHypTrapException(0x04);
    elsif HaveEL(EL3) && ELUsingAArch32(EL3) then
        PAR_NS = R[t2]:R[t];
    else
        PAR = R[t2]:R[t];
elsif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        PAR_NS = R[t2]:R[t];
    else
        PAR = R[t2]:R[t];
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        PAR_S = R[t2]:R[t];
    else
        PAR_NS = R[t2]:R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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 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' && (!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
        return 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
        return 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
        return PMCEID0;
elsif PSTATE.EL == EL3 then
    return PMCEID0;

```

3020/09/2021 1412:5237: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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 **Commoncommon** architectural events and **Commoncommon** microarchitectural events are implemented, or counted, using PMU events in the range 0x0020 to 0x003F.

For more information about the **Commoncommon** 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 [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 **Commoncommon** event (0x0020 + n).

For each bit:

ID<n>	Meaning
0b0	The Commoncommon event is not implemented, or not counted.
0b1	The Commoncommon event is implemented.

When the value of a bit in the field is 1, the corresponding **Commoncommon** event is implemented and counted.

Note

Arm recommends that if a **Commoncommon** 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 **Commoncommon** 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' && (!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
        return 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
        return 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
        return PMCEID1;
elsif PSTATE.EL == EL3 then
    return PMCEID1;

```

3020/09/2021 1412:5237: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCEID2, Performance Monitors Common Event Identification register 2

The PMCEID2 characteristics are:

Purpose

Defines which **Commoncommon** architectural events and **Commoncommon** microarchitectural events are implemented, or counted, using PMU events in the range 0x4000 to 0x401F.

For more information about the **Commoncommon** events and the use of the PMCEIDn registers see 'The PMU event number space and common events'.

Configuration

AArch32 System register PMCEID2 bits [31:0] are architecturally mapped to AArch64 System register [PMCEID0_EL0\[63:32\]](#).

AArch32 System register PMCEID2 bits [31:0] are architecturally mapped to External register [PMCEID2\[63:32\]](#).

This register is present only when AArch32 is supported and FEAT_PMUv3p1 is implemented. Otherwise, direct accesses to PMCEID2 are UNDEFINED.

Attributes

PMCEID2 is a 32-bit register.

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 **Commoncommon** event (0x4000 + n).

For each bit:

IDhi<n>	Meaning
0b0	The Commoncommon event is not implemented, or not counted.
0b1	The Commoncommon event is implemented.

When the value of a bit in the field is 1, the corresponding **Commoncommon** event is implemented and counted.

Note

Arm recommends that if a **Commoncommon** 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 **Commoncommon** 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 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' && (!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
        return 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
        return 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
        return PMCEID2;
elsif PSTATE.EL == EL3 then
    return PMCEID2;

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCEID3, Performance Monitors Common Event Identification register 3

The PMCEID3 characteristics are:

Purpose

Defines which **Commoncommon** architectural events and **Commoncommon** microarchitectural events are implemented, or counted, using PMU events in the range 0x4020 to 0x403F.

For more information about the **Commoncommon** events and the use of the PMCEIDn registers see 'The PMU event number space and common events'.

Configuration

AArch32 System register PMCEID3 bits [31:0] are architecturally mapped to AArch64 System register [PMCEID1_EL0\[63:32\]](#).

AArch32 System register PMCEID3 bits [31:0] are architecturally mapped to External register [PMCEID3\[63:32\]](#).

This register is present only when AArch32 is supported and FEAT_PMuV3p1 is implemented. Otherwise, direct accesses to PMCEID3 are UNDEFINED.

Attributes

PMCEID3 is a 32-bit register.

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 **Commoncommon** event (0x4020 + n).

For each bit:

IDhi<n>	Meaning
0b0	The Commoncommon event is not implemented, or not counted.
0b1	The Commoncommon event is implemented.

When the value of a bit in the field is 1, the corresponding **Commoncommon** event is implemented and counted.

Note

Arm recommends that if a **Commoncommon** 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 **Commoncommon** 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

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' && (!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
        return 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
        return 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
        return PMCEID3;
elsif PSTATE.EL == EL3 then
    return PMCEID3;

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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 [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	FZ0	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 != **0b000000000x00**:

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.

FZO	Meaning
0b0	Do not freeze on overflow.
0b1	Event counters do not count when PMOVSr [(N-1):0] is nonzero, where N is the value of HDCR .HPMN if EL2 is implemented, and PMCR .N otherwise.

If EL2 is implemented, then:

In the description of this field:

- This field affects the operation of event counters in the range [0 .. ([HDCR](#).HPMN-1)].
If EL2 is implemented and is using AArch32, PMN is [HDCR](#).HPMN.
- If [HDCR](#).HPMN is less than [PMCR](#).N:
 - This field does not affect the operation of event counters in the range [[HDCR](#).HPMN .. ([PMCR](#).N-1)].
 - The operation of this field ignores the values of [PMOVSr](#)[([PMCR](#).N-1):[HDCR](#).HPMN].

If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2](#).HPMN.

- This applies even when EL2 is disabled in the current Security state.

If EL2 is not implemented, PMN is [PMCR](#).N.

This field does not affect the operation of [PMCCNTR](#).

FZO	Meaning
0b0	Do not freeze on overflow.
0b1	Event counter PMEVCNTR <n> does not count when PMOVSr [(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 implemented, 0, this bit affects the operation of event counters in the range [0 .. (PMN-1)]. and [HDCR.HPMN](#) or [MDCR_EL2.HPMN](#) is less than PMCR.N, this bit does not affect the operation of event counters in the range [[HDCR.HPMN](#)..(PMCR.N-1)] or [[MDCR_EL2.HPMN](#)..(PMCR.N-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.

Note

The effect of [HDCR.HPMN](#) or [MDCR_EL2.HPMN](#) on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state. For more information, see the description of [HDCR.HPMN](#) or [MDCR_EL2.HPMN](#).

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 bit.
0b1	<p>Cycle counting by PMCCNTR is disabled in prohibited regions: disabled.</p> <p>When event counting by for counters in the range [0..(HDCR.HPMN-1)] or [0..(MDCR_EL2.HPMN-1)] is prohibited, cycle counting by PMCCNTR is disabled in prohibited regions: disabled.</p> <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 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.</p>

For more information see 'Prohibiting event 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 PE 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. ~~The possible values of this bit are:~~

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. ~~The effects of writing to this bit are:~~

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 Reset nall is event counters accessible in the range current of Exception affected event counters level, resets not each event counter including PMEVCNTR<n> PMCCNTR , to zero.

The effects of writing to this bit are:

- If EL2 is implemented and is 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)]. [HDCR.HPMN](#) or [MDCR_EL2.HPMN](#) is less than PMCR_EL0.N, a write of 1 to this bit does not reset event counters in the range [[HDCR.HPMN](#)..(PMCR.N-1)] or [[MDCR_EL2.HPMN](#)..(PMCR.N-1)].
- If EL2 is not implemented, EL2 is disabled in the current Security state, a write of 1 to this bit resets all the event counters. or [HDCR.HPMN](#) or [MDCR_EL2.HPMN](#) is equal to PMCR_EL0.N, 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](#).

In EL2 and EL3, a write of 1 to this bit resets all the event counters.

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.

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.

E	Meaning
0b0	All event counters in the range [0..(PMN-1)] and PMCCNTR , are disabled.
0b1	All event counters in the range [0..(PMN-1)] and PMCCNTR , are enabled by PMCNTENSET .

If EL2 is implemented then:

In the description of this field:

- If EL2 is using AArch32, PMN is [HDCR.HPMN](#).
If EL2 is implemented and is using AArch32, PMN is [HDCR.HPMN](#).
- If EL2 is using AArch64, PMN is [MDCR_EL2.HPMN](#).
If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2.HPMN](#).
- If PMN is less than PMCR.N, this bit does not affect the operation of event counters in the range [PMN..(PMCR.N-1)].
If EL2 is not implemented, PMN is PMCR.N.

If EL2 is not implemented, PMN is PMCR.N.

Note

The effect of [MDCR_EL2.HPMN](#) or [HDCR.HPMN](#) on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, regardless of whether EL2 is enabled in the current Security

state. For more information, see the description of [MDCR_EL2.HPMN](#) or [HDCR.HPMN](#).

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
        return 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
        return 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
        return PMCR;

```

```
elsif PSTATE.EL == EL3 then  
    return 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' && (!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];

```



```

        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMCR = R[t];
    elsif PSTATE.EL == EL3 then
        PMCR = R[t];

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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\[31:0\]](#).

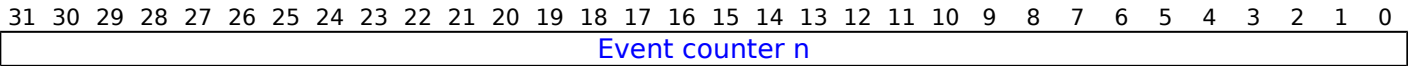
AArch32 System register PMEVCNTR<n> bits [31:0] are architecturally mapped to External register [PMEVCNTR<n>_EL0\[31:0\]](#).

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



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, 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>](#) 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>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b10:n[4:3]	n[2:0]

```

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.<ER,EN> == '00' then
        if EL2Enabled() && !ELUsingAArch32(EL2) && HCR_EL2.TGE == '1' then
            AArch64.AArch32SystemAccessTrap(EL2, 0x03);
        else
            AArch64.AArch32SystemAccessTrap(EL1, 0x03);
    elseif ELUsingAArch32(EL1) && PMUSERENR.<ER,EN> == '00' 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' && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HDFGRTR_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 HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        return PMEVCNTR[UInt(CRm<1:0>:opc2<2:0>)];
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
        return PMEVCNTR[UInt(CRm<1:0>:opc2<2:0>)];
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
        return PMEVCNTR[UInt(CRm<1:0>:opc2<2:0>)];
elseif PSTATE.EL == EL3 then
    return PMEVCNTR[UInt(CRm<1:0>:opc2<2:0>)];

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b10:n[4:3]	n[2:0]

```

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' && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HDBGWTR_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 HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVCNTR[UInt(CRm<1:0>:opc2<2:0>)] = 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
        PMEVCNTR[UInt(CRm<1:0>:opc2<2:0>)] = 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(CRm<1:0>:opc2<2:0>)] = R[t];
elseif PSTATE.EL == EL3 then
    PMEVCNTR[UInt(CRm<1:0>:opc2<2:0>)] = R[t];

```

3020/09/2021 14:12:5237: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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 [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 Software event must number program of this the field with an event that is counted supported by event the counter PE being programmed.~~ [PMEVCNTR<n>](#).

The ranges of event numbers allocated to each type of event are shown in 'Allocation of the PMU event number space'.

~~If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written. 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.~~

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 16-bit evtCount is implemented, for the range [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.
- ~~For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the evtCount field is UNKNOWN.~~

Note

UNPREDICTABLE means the event must not expose privileged information.

Arm recommends that for all values that represent reserved or unsupported events, 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.

~~Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.~~

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>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b11:n[4:3]	n[2:0]

```

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' && (!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 HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        return PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)];
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
        return PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)];
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
        return PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)];
elseif PSTATE.EL == EL3 then
    return PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)];

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1110	0b11:n[4:3]	n[2:0]

```

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(EL1) && HCR_EL2.<E2H,TGE> != '11' && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HDBGWTR_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 HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)] = 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 HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)] = 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
        PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)] = R[t];
elsif PSTATE.EL == EL3 then
    PMEVTYPER[UInt(CRm<1:0>:opc2<2:0>)] = R[t];

```

3020/09/2021 14:12:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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 AArch32 is supported 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								THWIDTH		BUS_WIDTH		BUS_WIDTH		BUS_SLOTS		BUS_SLOTSSLOTS		SLOTS													

Bits [31:24:20]

Reserved, RES0.

THWIDTH, bits [23:20]

PMEVTYPER<n>_EL0.TH width. Indicates implementation of the FEAT_PMUv3_TH feature, and, if implemented, the size of the **PMEVTYPER<n>_EL0.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>**.

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 Log₂(number of bytes), plus one. Defined values are:

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.

Access to this field is **RO**.

BUS_SLOTS, bits [15:8]

Bus count. The largest value by which the BUS_ACCESS event might increment **by** 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.

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 be as zero**. **RAZ**.

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;
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
            return PMMIR;
    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
                return PMMIR;
    elsif PSTATE.EL == EL3 then
        return PMMIR;

```

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PRRR, Primary Region Remap Register

The PRRR characteristics are:

Purpose

Controls the top level mapping of the TEX[0], C, and B memory region attributes.

Configuration

AArch32 System register PRRR bits [31:0] are architecturally mapped to AArch64 System register [MAIR_EL1\[31:0\]](#) when EL3 is not implemented or EL3 is using AArch64.

AArch32 System register PRRR bits [31:0] are architecturally mapped to AArch32 System register [MAIR0\[31:0\]](#) when EL3 is not implemented or EL3 is using AArch64.

AArch32 System register PRRR bits [31:0] (PRRR_S) are architecturally mapped to AArch32 System register [MAIR0\[31:0\]](#) (MAIR0_S) when EL3 is using AArch32.

AArch32 System register PRRR bits [31:0] (PRRR_NS) are architecturally mapped to AArch32 System register [MAIR0\[31:0\]](#) (MAIR0_NS) when EL3 is using AArch32.

This register is present only when AArch32 is supported. Otherwise, direct accesses to PRRR are UNDEFINED.

[MAIR0](#) and PRRR are the same register, with a different view depending on the value of [TTBCR.EAE](#):

- When it is set to 0, the register is as described in PRRR.
- When it is set to 1, the register is as described in [MAIR0](#).

Attributes

PRRR 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
NOS7	NOS6	NOS5	NOS4	NOS3	NOS2	NOS1	NOS0	RES0	NS1	NS0	DS1	DS0	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0											

NOS<n>, bit [n+24], for n = 7 to 0

Not Outer Shareable. NOS<n> is the Outer Shareable property for memory attributes n, if the region is mapped as Normal memory that is not Inner Non-cacheable, Outer Non-cacheable, and the appropriate PRRR.{NS0, NS1} field identifies the region as shareable. n is the value of the concatenation of the {TEX[0], C, B} bits from the translation table descriptor. The possible values of each NOS<n> field other than NOS6 are:

NOS<n>	Meaning
0b0	Memory region is Outer Shareable.
0b1	Memory region is Inner Shareable.

The value of this bit is ignored if the region is:

- Device memory
- Normal memory that is at least one of:
 - Inner Non-cacheable, Outer Non-cacheable.
 - Identified by the appropriate PRRR.{NS0, NS1} field as Non-shareable.

The meaning of the NOS6 field is IMPLEMENTATION DEFINED.

The reset behavior of this field is:

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

Bits [23:20]

Reserved, RES0.

NS1, bit [19]

Mapping of S = 1 attribute for Normal memory regions. This field is used in determining the Shareability of a memory region that is mapped to Normal memory and both:

- Is not Inner Non-cacheable, Outer Non-cacheable.
- Has the S bit in the translation table descriptor set to 1.

The possible values of this bit are:

NS1	Meaning
0b0	Region is Non-shareable.
0b1	Region is shareable. The value of the appropriate PRRR.NOS<n> field determines whether the region is Inner Shareable or Outer Shareable.

The reset behavior of this field is:

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

NS0, bit [18]

Mapping of S = 0 attribute for Normal memory regions. This field is used in determining the Shareability of a memory region that is mapped to Normal memory and both:

- Is not Inner Non-cacheable, Outer Non-cacheable.
- Has the S bit in the translation table descriptor set to 0.

The possible values of this bit are:

NS0	Meaning
0b0	Region is Non-shareable.
0b1	Region is shareable. The value of the appropriate PRRR.NOS<n> field determines whether the region is Inner Shareable or Outer Shareable.

The reset behavior of this field is:

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

DS1, bit [17]

Mapping of S = 1 attribute for Device memory. From Armv8, all types of Device memory are Outer Shareable, and therefore this bit is RES1.

The reset behavior of this field is:

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

DS0, bit [16]

Mapping of S = 0 attribute for Device memory. From Armv8, all types of Device memory are Outer Shareable, and therefore this bit is RES1.

The reset behavior of this field is:

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

TR<n>, bits [2n+1:2n], for n = 7 to 0

TR<n> is the primary TEX mapping for memory attributes n, and defines the mapped memory type for a region with attributes n. n is the value of the concatenation of the {TEX[0], C, B} bits from the translation table descriptor. The possible values for each field other than TR6 are:

TR<n>	Meaning
0b00	Device-nGnRnE memory
0b01	Device-nGnRE memory
0b10	Normal memory

The value 0b11 is reserved. The effect of programming a field to 0b11 is CONSTRAINED UNPREDICTABLE.

The meaning of the TR6 field is IMPLEMENTATION DEFINED.

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 PRRR

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	0b1010	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T10 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T10 == '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
        if TTBCR.EAE == '1' then
            return MAIR0_NS;
        else
            return PRRR_NS;
    else
        if TTBCR.EAE == '1' then
            return MAIR0;
        else
            return PRRR;
    elsif PSTATE.EL == EL2 then
        if HaveEL(EL3) && ELUsingAArch32(EL3) then
            if TTBCR.EAE == '1' then
                return MAIR0_NS;
            else
                return PRRR_NS;
        else
            if TTBCR.EAE == '1' then
                return MAIR0;
            else
                return PRRR;
    elsif PSTATE.EL == EL3 then
        if TTBCR.EAE == '1' then
            if SCR.NS == '0' then
                return MAIR0_S;
            else
                return MAIR0_NS;
        else
            if SCR.NS == '0' then
                return PRRR_S;
            else
                return PRRR_NS;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1010	0b0010	0b000

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T10 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T10 == '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
        if TTBCR.EAE == '1' then
            MAIR0_NS = R[t];
        else
            PRRR_NS = R[t];
        else
            if TTBCR.EAE == '1' then
                MAIR0 = R[t];
            else
                PRRR = R[t];
    elsif PSTATE.EL == EL2 then
        if HaveEL(EL3) && ELUsingAArch32(EL3) then
            if TTBCR.EAE == '1' then
                MAIR0_NS = R[t];
            else
                PRRR_NS = R[t];
        else
            if TTBCR.EAE == '1' then
                MAIR0 = R[t];
            else
                PRRR = 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 TTBCR.EAE == '1' then
                if SCR.NS == '0' then
                    MAIR0_S = R[t];
                else
                    MAIR0_NS = R[t];
            else
                if SCR.NS == '0' then
                    PRRR_S = R[t];
                else
                    PRRR_NS = R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbf36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

RMR, Reset Management Register

The RMR characteristics are:

Purpose

If EL1 or EL3 is the highest implemented Exception level and this register is implemented:

- A write to the register at the highest implemented Exception level can request a Warm reset.
- If the highest implemented Exception level can use AArch32 and AArch64, this register specifies the Execution state that the PE boots into on a Warm reset.

Configuration

AArch32 System register RMR bits [31:0] are architecturally mapped to AArch64 System register [RMR_EL1\[31:0\]](#) when the highest implemented Exception level is EL1.

AArch32 System register RMR bits [31:0] are architecturally mapped to AArch64 System register [RMR_EL3\[31:0\]](#) when EL3 is implemented.

This register is present only when AArch32 is supported. Otherwise, direct accesses to RMR are UNDEFINED.

Only implemented if EL1 or EL3 is the highest implemented Exception level. In this case:

- If the highest implemented Exception level can use AArch32 and AArch64 then this register must be implemented.
- If the highest implemented Exception level cannot use AArch64 then it is IMPLEMENTATION DEFINED whether the register is implemented.

Attributes

RMR 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																RR												AA64								

Bits [31:2]

Reserved, RES0.

RR, bit [1]

Reset Request. Setting this bit to 1 requests a Warm reset.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

AA64, bit [0]

When the highest implemented Exception level can use AArch64, determines which Execution state the PE boots into after a Warm reset:

AA64	Meaning
0b0	AArch32.
0b1	AArch64.

On coming out of the Warm reset, execution starts at the IMPLEMENTATION DEFINED reset vector address of the specified Execution state.

If the highest implemented Exception level cannot use AArch64 this bit is RAZ/WI.

When implemented as a RW field, this field resets to 0 on a Cold reset.

Accessing RMR

When EL3 is implemented, Arm deprecates accessing this register from any PE mode other than Monitor mode.

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	0b0000	0b010

```
if PSTATE.EL IN {EL1, EL3} && IsHighestEL(PSTATE.EL) then
    return RMR;
else
    UNDEFINED;
```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b1100	0b0000	0b010

```
if PSTATE.EL == IN EL0{EL1, then
    UNDEFINED;
elseif EL3} PSTATE.EL && == EL1 then
    if IsHighestEL(EL1PSTATE.EL) then
        RMR = R[t];
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    if CP15SDISABLE == HIGH then
        UNDEFINED;
    elseif CP15SDISABLE2 == HIGH then
        UNDEFINED;
    else
        RMR = R[t];
```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

SCR, Secure Configuration Register

The SCR characteristics are:

Purpose

When EL3 is implemented and can use AArch32, defines the configuration of the current Security state. It specifies:

- The Security state, either Secure or Non-secure.
- What mode the PE branches to if an IRQ, FIQ, or External abort occurs.
- Whether the PSTATE.F or PSTATE.A bits can be modified when SCR.NS==1.

Configuration

AArch32 System register SCR bits [31:0] can be mapped to AArch64 System register [SCR_EL3\[31:0\]](#), but this is not architecturally mandated.

This register is present only when AArch32 is supported. Otherwise, direct accesses to SCR are UNDEFINED.

Attributes

SCR 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																TERR	RES0	TWEL	TWEL	RES0	SIFH	CE	SCDn	ETAW	FW	EA	FIQ	IRQ	NS		

Bits [31:16]

Reserved, RES0.

TERR, bit [15]

When FEAT_RAS is implemented:

Trap Error record accesses. Generate a Monitor Trap exception on accesses to the following registers from modes other than Monitor mode:

[ERRIDR](#), [ERRSELR](#), [ERXADDR](#), [ERXADDR2](#), [ERXCTLR](#), [ERXCTLR2](#), [ERXFR](#), [ERXFR2](#), [ERXMISC0](#), [ERXMISC1](#), [ERXMISC2](#), [ERXMISC3](#), and [ERXSTATUS](#). When FEAT_RASv1p1 is implemented, [ERXMISC4](#), [ERXMISC5](#), [ERXMISC6](#), [ERXMISC7](#).

TERR	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Accesses to the specified registers from modes other than Monitor mode generate a Monitor Trap exception.

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.

Bit [14]

Reserved, RES0.

TWE, bit [13]

Traps WFE instructions to Monitor mode.

TWE	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any attempt to execute a WFE instruction in any mode other than Monitor mode is trapped to Monitor mode, if the instruction would otherwise have caused the PE to enter a low-power state and the attempted execution does not generate an exception that is taken to EL1 or EL2 by SCTLR.nTWE or HCR.TWE . Any exception that is taken to EL1 or to EL2 has priority over this trap.

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 EL3, this field resets to 0.

TWI, bit [12]

Traps WFI instructions to Monitor mode.

TWI	Meaning
0b0	This control does not cause any instructions to be trapped.
0b1	Any attempt to execute a WFI instruction in any mode other than Monitor mode is trapped to Monitor mode, if the instruction would otherwise have caused the PE to enter a low-power state and the attempted execution does not generate an exception that is taken to EL1 or EL2 by SCTLR.nTWI or HCR.TWI . Any exception that is taken to EL1 or to EL2 has priority over this trap.

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 EL3, this field resets to 0.

Bits [11:10]

Reserved, RES0.

SIF, bit [9]

Secure instruction fetch. When the PE is in Secure state, this bit disables instruction fetch from Non-secure memory. **The possible values for this bit are:**

SIF	Meaning
0b0	Secure state instruction fetches from Non-secure memory are permitted.
0b1	Secure state instruction fetches from Non-secure memory are not permitted.

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 EL3, this field resets to 0.

HCE, bit [8]

Hypervisor Call instruction enable. If EL2 is implemented, enables execution of HVC instructions at Non-secure EL1 and EL2.

HCE	Meaning
0b0	HVC instructions are: <ul style="list-style-type: none"> UNDEFINED at Non-secure EL1. The Undefined Instruction exception is taken from PL1 to PL1. UNPREDICTABLE at EL2. Behavior is one of the following: <ul style="list-style-type: none"> The instruction is UNDEFINED. The instruction executes as a NOP.
0b1	HVC instructions are enabled at Non-secure EL1 and EL2.

Note

HVC instructions are always UNDEFINED at EL0 and in Secure state.

If EL2 is not implemented, this bit is RES0 and HVC is UNDEFINED.

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.

SCD, bit [7]

Secure Monitor Call disable. Disables SMC instructions.

SCD	Meaning
0b0	SMC instructions are enabled.
0b1	In Non-secure state, SMC instructions are UNDEFINED. The Undefined Instruction exception is taken from the current Exception level to the current Exception level. In Secure state, behavior is one of the following: <ul style="list-style-type: none"> The instruction is UNDEFINED. The instruction executes as a NOP.

Note

SMC instructions are always UNDEFINED at PL0.

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.

nET, bit [6]

Not Early Termination. This bit disables early termination.

nET	Meaning
0b0	Early termination permitted. Execution time of data operations can depend on the data values.
0b1	Disable early termination. The number of cycles required for data operations is forced to be independent of the data values.

This IMPLEMENTATION DEFINED mechanism can disable data dependent timing optimizations from multiplies and data operations. It can provide system support against information leakage that might be exploited by timing correlation types of attack.

On implementations that do not support early termination or do not support disabling early termination, this bit is RES0.

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.

AW, bit [5]

When the value of SCR.EA is 1 and the value of [HCR](#).AMO is 0, this bit controls whether PSTATE.A masks an External abort taken from Non-secure state.

AW	Meaning
0b0	External aborts taken from Non-secure state are not masked by PSTATE.A, and are taken to EL3. External aborts taken from Secure state are masked by PSTATE.A.
0b1	External aborts taken from either Security state are masked by PSTATE.A. When PSTATE.A is 0, the abort is taken to EL3.

When SCR.EA is 0 or [HCR](#).AMO is 1, this bit has no effect.

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.

FW, bit [4]

When the value of SCR.FIQ is 1 and the value of [HCR](#).FMO is 0, this bit controls whether PSTATE.F masks an FIQ interrupt taken from Non-secure state.

FW	Meaning
0b0	An FIQ taken from Non-secure state is not masked by PSTATE.F, and is taken to EL3. An FIQ taken from Secure state is masked by PSTATE.F.
0b1	An FIQ taken from either Security state is masked by PSTATE.F. When PSTATE.F is 0, the FIQ is taken to EL3.

When SCR.FIQ is 0 or [HCR](#).FMO is 1, this bit has no effect.

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.

EA, bit [3]

External Abort handler. This bit controls which mode takes External aborts and SError interrupt exceptions.

EA	Meaning
0b0	External aborts taken to Abort mode.
0b1	External aborts taken to Monitor mode.

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.

FIQ, bit [2]

FIQ handler. This bit controls which mode takes FIQ exceptions.

FIQ	Meaning
0b0	FIQs taken to FIQ mode.
0b1	FIQs taken to Monitor mode.

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.

IRQ, bit [1]

IRQ handler. This bit controls which mode takes IRQ exceptions.

IRQ	Meaning
0b0	IRQs taken to IRQ mode.
0b1	IRQs taken to Monitor mode.

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.

NS, bit [0]

Non-secure bit. Except when the PE is in Monitor mode, this bit determines the Security state of the PE:

NS	Meaning
0b0	PE is in Secure state.
0b1	PE is in Non-secure state.

If the [HCR.TGE](#) bit is set, an attempt to change from a Secure PL1 mode to a Non-secure EL1 mode by changing the SCR.NS bit from 0 to 1 results in the SCR.NS bit remaining as 0.

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 SCR

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	0b0001	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 !ELUsingAArch32(EL2) && SCR_EL3.<NS,EEL2> == '01' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif !ELUsingAArch32(EL3) && SCR_EL3.NS == '0' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return SCR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0001	0b0001	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 !ELUsingAArch32(EL2) && SCR_EL3.<NS,EEL2> == '01' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif !ELUsingAArch32(EL3) && SCR_EL3.NS == '0' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SCR = R[t];

```

3020/09/2021 14:12:5237: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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 AArch32 is supported. 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	A	FE	TRE	RES0	EE	RES0	SPAN	RES1	RES0	UWXN	WXN	nTWE	RES0	nTWI	RES0	V	I	RES1	EnRCTX	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. ~~The possible values of this bit are:~~

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. ~~The possible values of this bit are:~~

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.

~~The possible values of this bit are:~~

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. **The possible values of this bit are:**

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. ~~The possible values of this bit are:~~

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_SPECREG is implemented:

Enable EL0 access to the AArch32 CFPRCTX, DVPRCTX, and CPPRCTX instructions.

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.

In 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
        return SCTLR_NS;
    else
        return SCTLR;
elsif PSTATE.EL == EL2 then
    if HaveEL(EL3) && ELUsingAArch32(EL3) then
        return SCTLR_NS;
    else
        return SCTLR;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' then
        return SCTLR_S;
    else
        return 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];

```

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

SDCR, Secure Debug Control Register

The SDCR characteristics are:

Purpose

Provides EL3 configuration options for self-hosted debug, trace, and the Performance Monitors Extension.

Configuration

AArch32 System register SDCR bits [31:0] can be mapped to AArch64 System register [MDCR_EL3\[31:0\]](#), but this is not architecturally mandated.

This register is present only when AArch32 is supported. Otherwise, direct accesses to SDCR are UNDEFINED.

Attributes

SDCR 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	MTPME	TDCC	RES0	SCCD	RES0	EPMA	EDAD	TTRF	STE	SPME	RES0	SPD	RES0																		

Bits [31:29]

Reserved, RES0.

MTPME, bit [28] When FEAT_MTPMU is implemented:

Multi-threaded PMU Enable. Enables use of the [PMEVTYPER<n>](#).MT bits.

MTPME	Meaning
0b0	FEAT_MTPMU is disabled. The Effective value of PMEVTYPER<n> .MT is 0.
0b1	PMEVTYPER<n> .MT bits not affected by this bit.

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 bit is 0.

The reset behavior of this field is:

- On a Cold reset, in a system where the PE resets into EL3, 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 in modes other than Monitor mode to Monitor mode.

TDCC	Meaning
0b0	This control does not cause any register accesses to be trapped.
0b1	Accesses to the DCC registers in modes other than Monitor mode generate a Monitor Trap exception, 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:

- [DBGDTRRXext](#), [DBGDTRTXext](#), [DBGDSCRint](#), [DBGDCCINT](#), and, when the PE is in Non-debug state, [DBGDTRRXint](#) and [DBGDTRTXint](#).

When the PE is in Debug state, SDCR.TDCC does not trap any accesses to:

- [DBGDTRRXint](#) and [DBGDTRTXint](#).

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 [26:24]

Reserved, RES0.

SCCD, bit [23]

When FEAT_PMUv3p5 is implemented:

Secure Cycle Counter Disable. Prohibits [PMCCNTR](#) from counting in Secure state.

SCCD	Meaning
0b0	Cycle counting by PMCCNTR is not affected by this mechanism.
0b1	Cycle counting by PMCCNTR 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, in a system where the PE resets into EL3, this field resets to 0.

Otherwise:

Reserved, RES0.

Bit [22]

Reserved, RES0.

EPMAD, bit [21]

When FEAT_Debugv8p4 is implemented and FEAT_PMUv3 is implemented:

External Performance Monitors Non-secure access disable. Controls Non-secure access to Performance Monitors registers by an external debugger.

EPMAD	Meaning
0b0	Non-secure access to the Performance Monitors registers from an external debugger is permitted.
0b1	Non-secure access to the Performance Monitors registers from an external debugger is not permitted.

If the Performance Monitors Extension does not support external debug interface accesses, this bit is RES0.

Otherwise, if EL3 is not implemented and the Effective value of [SCR.NS](#) is 0, then the Effective value of this field 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 0.

When FEAT_PMuV3 is implemented:

External Performance Monitors access disable. Controls access to Performance Monitors registers by an external debugger.

EPMAD	Meaning
0b0	Access to Performance Monitors registers from an external debugger is permitted.
0b1	Access to Performance Monitors registers from an external debugger is not permitted, unless overridden by the IMPLEMENTATION DEFINED authentication interface.

If the Performance Monitors Extension does not support external debug interface accesses, this bit is RES0.

Otherwise, if EL3 is not implemented and the Effective value of [SCR.NS](#) is 0, then the Effective value of this field 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 0.

Otherwise:

Reserved, RES0.

EDAD, bit [20]

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 an external debugger is permitted.
0b1	Non-secure access to breakpoint registers, watchpoint registers, and OSLAR_EL1 from an external debugger is not permitted.

If EL3 is not implemented and the Effective value of [SCR.NS](#) is 0, then the Effective value of this field 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 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 from an external debugger is permitted.
0b1	Access to breakpoint registers, watchpoint registers, and OSLAR_EL1 from an external debugger is not permitted, unless overridden by the IMPLEMENTATION DEFINED authentication interface.

If EL3 is not implemented and the Effective value of [SCR.NS](#) is 0, then the Effective value of this field 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 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 an external debugger is permitted.
0b1	Access to breakpoint registers 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.NS](#) is 0, then the Effective value of this field 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 0.

TTRF, bit [19]

When FEAT_TRF is implemented:

Trap Trace Filter controls. Controls whether accesses in modes other than Monitor mode to the trace filter control registers generate a Monitor Trap exception.

TTRF	Meaning
0b0	Accesses to HTRECR and TRECR are not affected by this control bit.
0b1	When not in Monitor mode, accesses to HTRECR and TRECR generate a Monitor Trap exception, unless the access generates a higher priority exception.

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.

STE, bit [18]

When FEAT_TRF is implemented:

Secure Trace Enable. This bit enables tracing in Secure state and controls the level of authentication required by an external debugger to enable external tracing.

STE	Meaning
0b0	Trace is 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.NS](#) is 0, the PE behaves as if this bit is set to 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 0.

Otherwise:

Reserved, RES0.

SPME, bit [17]

When FEAT_PMUv3 is implemented and FEAT_Debugv8p2 is implemented:

Secure Performance Monitors Enable. Controls event counting in Secure state.

SPME	Meaning
0b0	Event counting is prohibited in Secure state. If PMCR.DP is 1, PMCCNTR is disabled in Secure state. Otherwise, PMCCNTR is not affected by this mechanism.
0b1	Event counting and PMCCNTR are not affected by this mechanism.

This field affects the operation of all event counters in Secure state, and if [PMCR.DP](#) is 1, the operation of [PMCCNTR](#) in Secure state. When [PMCR.DP](#) is 0, [PMCCNTR](#) is not affected by this field.

If EL3 is not implemented and the Effective value of [SCR.NS](#) is 0, then the Effective value of this field 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 0.

When FEAT_PMUv3 is implemented:

Secure Performance Monitors Enable. Controls event counting in Secure state.

SPME	Meaning
0b0	If ExternalSecureNoninvasiveDebugEnabled() is FALSE, event counting is prohibited in Secure state, and if PMCR.DP is 1, PMCCNTR is disabled in Secure state.
0b1	Event counting and PMCCNTR are not affected by this mechanism.

If ExternalSecureNoninvasiveDebugEnabled() is TRUE, the event counters and [PMCCNTR](#) are not affected by this field.

Otherwise, this field affects the operation of all event counters in Secure state, and if [PMCR.DP](#) is 1, the operation of [PMCCNTR](#) in Secure state. When [PMCR.DP](#) is 0, [PMCCNTR](#) is not affected by this field.

If EL3 is not implemented and the Effective value of [SCR.NS](#) is 0, then the Effective value of this field 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 0.

Otherwise:

Reserved, RES0.

Bit [16]

Reserved, RES0.

SPD, bits [15:14]

AArch32 Secure self-hosted Privileged Debug. Enables or disables debug exceptions from EL3, other than Breakpoint Instruction exceptions.

SPD	Meaning
0b00	Legacy mode. Debug exceptions from EL3 are enabled by the authentication interface.
0b10	Secure privileged debug disabled. Debug exceptions from EL3 are disabled.
0b11	Secure privileged debug enabled. Debug exceptions from EL3 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 in Non-secure state.

If debug exceptions from EL3 are enabled, then debug exceptions from Secure EL0 are also enabled.

Otherwise, debug exceptions from Secure EL0 are enabled only if the value of [SDER.SUIDEN](#) is 1.

If EL3 is not implemented and the Effective value of [SCR.NS](#) is 0, then the Effective value of this field is 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 0.

Bits [13:0]

Reserved, RES0.

Accessing SDCR

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	0b0011	0b001

```

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 !ELUsingAArch32(EL2) && SCR_EL3.<NS,EEL2> == '01' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif !ELUsingAArch32(EL3) && SCR_EL3.NS == '0' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return SDCR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0001	0b0011	0b001

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && !ELUsingAArch32(EL2) && HSTR_EL2.T1 == '1' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif EL2Enabled() && ELUsingAArch32(EL2) && HSTR.T1 == '1' then
        AArch32.TakeHypTrapException(0x03);
    elseif !ELUsingAArch32(EL2) && SCR_EL3.<NS,EEL2> == '01' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elseif !ELUsingAArch32(EL3) && SCR_EL3.NS == '0' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    if SCR.NS == '0' && CP15SDISABLE2 == HIGH then
        UNDEFINED;
    else
        SDCR = R[t];

```

3020/09/2021 1412:5236; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

SDER, Secure Debug Enable Register

The SDER characteristics are:

Purpose

Controls invasive and non-invasive debug in the Secure EL0 mode.

Configuration

AArch32 System register SDER bits [31:0] are architecturally mapped to AArch64 System register [SDER32_EL2\[31:0\]](#) when EL2 is implemented and FEAT_SEL2 is implemented.

AArch32 System register SDER bits [31:0] are architecturally mapped to AArch64 System register [SDER32_EL3\[31:0\]](#) when EL3 is implemented.

This register is present only when (EL3 is implemented and EL3 is capable of using AArch32) or (EL1 is capable of using AArch32 and Secure EL1 is implemented). Otherwise, direct accesses to SDER are UNDEFINED.

This register is ignored by the PE when one or more of the following are true:

- The PE is in Non-secure state.
- EL1 is using AArch64.

Attributes

SDER 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																SUNIDEN		SUIDEN													

Bits [31:2]

Reserved, RES0.

SUNIDEN, bit [1]

Secure User Non-Invasive Debug Enable.

SUNIDEN	Meaning
0b0	This bit does not affect Performance Monitors event counting at Secure EL0
0b1	If EL3 or EL1 is using AArch32, Performance Monitors event counting is allowed in Secure EL0.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

SUIDEN, bit [0]

Secure User Invasive Debug Enable.

SUIDEN	Meaning
0b0	This bit does not affect the generation of debug exceptions at Secure EL0.
0b1	If EL3 or EL1 is using AArch32, debug exceptions from Secure EL0 are enabled.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing SDER

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	0b0001	0b001

```

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 !IsSecure() then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        return SDER;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return SDER;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0001	0b0001	0b001


```

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 !IsSecure() then
        UNDEFINED;
    elsif EL2Enabled() && !ELUsingAArch32(EL2) && MDCR_EL2.<TDE,TDA> != '00' then
        AArch64.AArch32SystemAccessTrap(EL2, 0x03);
    elsif HaveEL(EL3) && !ELUsingAArch32(EL3) && MDCR_EL3.TDA == '1' then
        AArch64.AArch32SystemAccessTrap(EL3, 0x03);
    else
        SDER = R[t];
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    if SCR.NS == '0' && CP15SDISABLE2 == HIGH then
        UNDEFINED;
    else
        SDER = R[t];

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd36e47856e443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

VMPIDR, Virtualization Multiprocessor ID Register

The VMPIDR characteristics are:

Purpose

Holds the value of the Virtualization Multiprocessor ID. This is the value returned by Non-secure EL1 reads of [MPIDR](#).

Configuration

AArch32 System register VMPIDR bits [31:0] are architecturally mapped to AArch64 System register [VMPIDR_EL2\[31:0\]](#).

This register is present only when AArch32 is supported. Otherwise, direct accesses to VMPIDR are UNDEFINED.

If EL2 is not implemented but EL3 is implemented, this register takes the value of the [MPIDR](#).

Attributes

VMPIDR 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
M	U					RES0	MT																								

M, bit [31]

Indicates whether this implementation includes the functionality introduced by the [Arm v7-ARMv7](#) Multiprocessing Extensions. ~~The possible values of this bit are:~~

M	Meaning
0b0	This implementation does not include the Arm v7-ARMv7 Multiprocessing Extensions functionality.
0b1	This implementation includes the Arm v7-ARMv7 Multiprocessing Extensions functionality.

~~Access From to Arm v8~~ this ~~field~~ bit is RES1.

U, bit [30]

Indicates a Uniprocessor system, as distinct from PE 0 in a multiprocessor system. ~~The possible values of this bit are:~~

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, in a system where the PE resets into EL2 or EL3, this field resets to the value in [MPIDR.U](#).

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. **The possible values of this bit are:**

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, in a system where the PE resets into EL2 or EL3, this field resets to the value in [MPIDR.MT](#).

Aff2, bits [23:16]

Affinity level 2. See the description of Aff0 for more information.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to the value in [MPIDR.Aff2](#).

Aff1, bits [15:8]

Affinity level 1. See the description of Aff0 for more information.

The reset behavior of this field is:

- On a Warm reset, in a system where the PE resets into EL2 or EL3, this field resets to the value in [MPIDR.Aff1](#).

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, in a system where the PE resets into EL2 or EL3, this field resets to the value in [MPIDR.Aff0](#).

Accessing VMPIDR

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	0b0000	0b0000	0b101

```

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);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return VMPIDR;
elsif PSTATE.EL == EL3 then
    if !HaveEL(EL2) then
        return MPIDR;
    elsif SCR.NS == '0' then
        UNDEFINED;
    else
        return VMPIDR;

```

MCR{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b100	0b0000	0b0000	0b101

```

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);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    VMPIDR = R[t];
elsif PSTATE.EL == EL3 then
    if !HaveEL(EL2) then
        //no operation
    elsif SCR.NS == '0' then
        UNDEFINED;
    else
        VMPIDR = R[t];

```

MRC{<c>}{<q>} <coproc>, {#}<opc1>, <Rt>, <CRn>, <CRm>{, {#}<opc2>}

coproc	opc1	CRn	CRm	opc2
0b1111	0b000	0b0000	0b0000	0b101

```

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) then
        return VMPIDR_EL2<31:0>;
    elsif EL2Enabled() && ELUsingAArch32(EL2) then
        return VMPIDR;
    else
        return MPIDR;
elsif PSTATE.EL == EL2 then
    return MPIDR;
elsif PSTATE.EL == EL3 then
    return MPIDR;

```

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd5b36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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/VMRS](#)

For AArch64

- [AT](#)
- [BRB](#)
- [CFP](#)
- [CPP](#)
- [DC](#)
- [DVP](#)
- [IC](#)
- [MRS/MSR](#)
- [TLBI](#)

Registers and operations in AArch32

Accessed using MCR/MRC:

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1110	0b000	0b0000	0b0000	0b000	DBGDIDR	Debug ID Register
0b1110	0b000	0b0000	0b0000	0b010	DBGDTRRText	Debug OS Local Data Transfer Register, Receive, External View
0b1110	0b000	0b0000	0b0001	0b000	DBGDSCRint	Debug Status and Control Register, Internal View
0b1110	0b000	0b0000	0b0010	0b000	DBGDCCINT	DCC Interrupt Enable Register
0b1110	0b000	0b0000	0b0010	0b010	DBGDSCRext	Debug Status and Control Register, External View
0b1110	0b000	0b0000	0b0011	0b010	DBGDTRTXtext	Debug OS Local Data Transfer Register, Transmit
0b1110	0b000	0b0000	0b0101	0b000	DBGDTRRXint	Debug Data Transfer Register, Receive
0b1110	0b000	0b0000	0b0101	0b000	DBGDTRTXint	Debug Data Transfer Register, Transmit

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1110	0b000	0b0000	0b0110	0b000	DBGWFAR	Debug Watchpoint Fault Address Register
0b1110	0b000	0b0000	0b0110	0b010	DBGOSECCR	Debug OS Lock Exception Catch Control Register
0b1110	0b000	0b0000	0b0111	0b000	DBGVCR	Debug Vector Catch Register
0b1110	0b000	0b0000	n[3:0]	0b100	DBGBVR<n>	Debug Breakpoint Value Register
0b1110	0b000	0b0000	n[3:0]	0b101	DBGBCR<n>	Debug Breakpoint Control Registers
0b1110	0b000	0b0000	n[3:0]	0b110	DBGWVR<n>	Debug Watchpoint Value Register
0b1110	0b000	0b0000	n[3:0]	0b111	DBGWCR<n>	Debug Watchpoint Control Registers
0b1110	0b000	0b0001	0b0000	0b000	DBGDRAR	Debug ROM Address Register
0b1110	0b000	0b0001	0b0000	0b100	DBGOSLAR	Debug OS Lock Access Register
0b1110	0b000	0b0001	0b0001	0b100	DBGOSLSR	Debug OS Lock Status Register
0b1110	0b000	0b0001	0b0011	0b100	DBGOSDLR	Debug OS Double Lock Register
0b1110	0b000	0b0001	0b0100	0b100	DBGPRCR	Debug Power Control Register
0b1110	0b000	0b0001	n[3:0]	0b001	DBGBXVR<n>	Debug Breakpoint Extended Value Registers
0b1110	0b000	0b0010	0b0000	0b000	DBGDSAR	Debug Self Address Register
0b1110	0b000	0b0111	0b0000	0b111	DBGDEVID2	Debug Device ID register 2
0b1110	0b000	0b0111	0b0001	0b111	DBGDEVID1	Debug Device ID register 1
0b1110	0b000	0b0111	0b0010	0b111	DBGDEVID	Debug Device ID register 0
0b1110	0b000	0b0111	0b1000	0b110	DBGCLAIMSET	Debug CLAIM Tag Set register
0b1110	0b000	0b0111	0b1001	0b110	DBGCLAIMCLR	Debug CLAIM Tag Clear register
0b1110	0b000	0b0111	0b1110	0b110	DBGAUTHSTATUS	Debug Authentication Status register
0b1110	0b111	0b0000	0b0000	0b000	JIDR	Jazelle ID Register

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1110	0b111	0b0001	0b0000	0b000	JOSCR	Jazelle OS Control Register
0b1110	0b111	0b0010	0b0000	0b000	JMCR	Jazelle Main Configuration Register
0b1111	0b000	0b0000	0b0000	0b000	MIDR	Main ID Register
0b1111	0b000	0b0000	0b0000	0b001	CTR	Cache Type Register
0b1111	0b000	0b0000	0b0000	0b010	TCMTR	TCM Type Register
0b1111	0b000	0b0000	0b0000	0b011	TLBTR	TLB Type Register
0b1111	0b000	0b0000	0b0000	0b101	MPIDR	Multiprocessor Affinity Register
0b1111	0b000	0b0000	0b0000	0b110	REVIDR	Revision ID Register
0b1111	0b000	0b0000	0b0001	0b000	ID_PFR0	Processor Feature Register 0
0b1111	0b000	0b0000	0b0001	0b001	ID_PFR1	Processor Feature Register 1
0b1111	0b000	0b0000	0b0001	0b010	ID_DFR0	Debug Feature Register 0
0b1111	0b000	0b0000	0b0001	0b011	ID_AFR0	Auxiliary Feature Register 0
0b1111	0b000	0b0000	0b0001	0b100	ID_MMFR0	Memory Model Feature Register 0
0b1111	0b000	0b0000	0b0001	0b101	ID_MMFR1	Memory Model Feature Register 1
0b1111	0b000	0b0000	0b0001	0b110	ID_MMFR2	Memory Model Feature Register 2
0b1111	0b000	0b0000	0b0001	0b111	ID_MMFR3	Memory Model Feature Register 3
0b1111	0b000	0b0000	0b0010	0b000	ID_ISAR0	Instruction Set Attribute Register 0
0b1111	0b000	0b0000	0b0010	0b001	ID_ISAR1	Instruction Set Attribute Register 1
0b1111	0b000	0b0000	0b0010	0b010	ID_ISAR2	Instruction Set Attribute Register 2
0b1111	0b000	0b0000	0b0010	0b011	ID_ISAR3	Instruction Set Attribute Register 3
0b1111	0b000	0b0000	0b0010	0b100	ID_ISAR4	Instruction Set Attribute Register 4
0b1111	0b000	0b0000	0b0010	0b101	ID_ISAR5	Instruction Set Attribute Register 5

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1111	0b000	0b0000	0b0010	0b110	ID_MMFR4	Memory Mode Feature Register 4
0b1111	0b000	0b0000	0b0010	0b111	ID_ISAR6	Instruction Set Attribute Register 6
0b1111	0b000	0b0000	0b0011	0b100	ID_PFR2	Processor Feature Register 2
0b1111	0b000	0b0000	0b0011	0b101	ID_DFR1	Debug Feature Register 1
0b1111	0b000	0b0000	0b0011	0b110	ID_MMFR5	Memory Mode Feature Register 5
0b1111	0b000	0b0001	0b0000	0b000	SCTLR	System Control Register
0b1111	0b000	0b0001	0b0000	0b001	ACTLR	Auxiliary Control Register
0b1111	0b000	0b0001	0b0000	0b010	CPACR	Architectural Feature Access Control Register
0b1111	0b000	0b0001	0b0000	0b011	ACTLR2	Auxiliary Control Register 2
0b1111	0b000	0b0001	0b0001	0b000	SCR	Secure Configuration Register
0b1111	0b000	0b0001	0b0001	0b001	SDER	Secure Debug Enable Register
0b1111	0b000	0b0001	0b0001	0b010	NSACR	Non-Secure Access Control Register
0b1111	0b000	0b0001	0b0010	0b001	TRFCR	Trace Filter Control Register
0b1111	0b000	0b0001	0b0011	0b001	SDCR	Secure Debug Control Register
0b1111	0b000	0b0010	0b0000	0b000	TTBR0	Translation Table Base Register 0
0b1111	0b000	0b0010	0b0000	0b001	TTBR1	Translation Table Base Register 1
0b1111	0b000	0b0010	0b0000	0b010	TTBCR	Translation Table Base Control Register
0b1111	0b000	0b0010	0b0000	0b011	TTBCR2	Translation Table Base Control Register 2
0b1111	0b000	0b0011	0b0000	0b000	DACR	Domain Access Control Register
0b1111	0b000	0b0100	0b0110	0b000	ICC_PMR	Interrupt Controller Interrupt

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
						Priority Mask Register
0b1111	0b000	0b0101	0b0000	0b000	DFSR	Data Fault Status Register
0b1111	0b000	0b0101	0b0000	0b001	IFSR	Instruction Fault Status Register
0b1111	0b000	0b0101	0b0001	0b000	ADFSR	Auxiliary Data Fault Status Register
0b1111	0b000	0b0101	0b0001	0b001	AIFSR	Auxiliary Instruction Fault Status Register
0b1111	0b000	0b0101	0b0011	0b000	ERRIDR	Error Record Register
0b1111	0b000	0b0101	0b0011	0b001	ERRSELR	Error Record Select Register
0b1111	0b000	0b0101	0b0100	0b000	ERXFR	Selected Error Record Feature Register
0b1111	0b000	0b0101	0b0100	0b001	ERXCTLR	Selected Error Record Control Register
0b1111	0b000	0b0101	0b0100	0b010	ERXSTATUS	Selected Error Record Primary Status Register
0b1111	0b000	0b0101	0b0100	0b011	ERXADDR	Selected Error Record Address Register
0b1111	0b000	0b0101	0b0100	0b100	ERXFR2	Selected Error Record Feature Register 2
0b1111	0b000	0b0101	0b0100	0b101	ERXCTLR2	Selected Error Record Control Register 2
0b1111	0b000	0b0101	0b0100	0b111	ERXADDR2	Selected Error Record Address Register 2
0b1111	0b000	0b0101	0b0101	0b000	ERXMISC0	Selected Error Record Miscellaneous Register 0
0b1111	0b000	0b0101	0b0101	0b001	ERXMISC1	Selected Error Record Miscellaneous Register 1
0b1111	0b000	0b0101	0b0101	0b010	ERXMISC4	Selected Error Record Miscellaneous Register 4
0b1111	0b000	0b0101	0b0101	0b011	ERXMISC5	Selected Error Record Miscellaneous Register 5
0b1111	0b000	0b0101	0b0101	0b100	ERXMISC2	Selected Error Record Miscellaneous Register 2
0b1111	0b000	0b0101	0b0101	0b101	ERXMISC3	Selected Error Record Miscellaneous Register 3

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
						Miscellaneous Register 3
0b1111	0b000	0b0101	0b0101	0b110	ERXMISC6	Selected Error Record Miscellaneous Register 6
0b1111	0b000	0b0101	0b0101	0b111	ERXMISC7	Selected Error Record Miscellaneous Register 7
0b1111	0b000	0b0110	0b0000	0b000	DFAR	Data Fault Address Register
0b1111	0b000	0b0110	0b0000	0b010	IFAR	Instruction Fault Address Register
0b1111	0b000	0b0111	0b0001	0b000	ICIALLUIS	Instruction Cache Invalidate All PoU, Inner Shareable
0b1111	0b000	0b0111	0b0001	0b110	BPIALLIS	Branch Predictor Invalidate All, Inner Shareable
0b1111	0b000	0b0111	0b0011	0b100	CFPRCTX	Control Flow Prediction Restriction by Context
0b1111	0b000	0b0111	0b0011	0b101	DVPRCTX	Data Value Prediction Restriction by Context
0b1111	0b000	0b0111	0b0011	0b111	CPPRCTX	Cache Prefetch Prediction Restriction by Context
0b1111	0b000	0b0111	0b0100	0b000	PAR	Physical Address Register
0b1111	0b000	0b0111	0b0101	0b000	ICIALLU	Instruction Cache Invalidate All PoU
0b1111	0b000	0b0111	0b0101	0b001	ICIMVAU	Instruction Cache line Invalidate by VA to PoU
0b1111	0b000	0b0111	0b0101	0b100	CP15ISB	Instruction Synchronization Barrier System instruction
0b1111	0b000	0b0111	0b0101	0b110	BPIALL	Branch Predictor Invalidate All
0b1111	0b000	0b0111	0b0101	0b111	BPIMVA	Branch Predictor Invalidate by VA
0b1111	0b000	0b0111	0b0110	0b001	DCIMVAC	Data Cache line Invalidate by VA to PoC

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1111	0b000	0b0111	0b0110	0b010	DCISW	Data Cache li Invalidate by Set/Way
0b1111	0b000	0b0111	0b1000	0b000	ATS1CPR	Address Translate Sta 1 Current sta PL1 Read
0b1111	0b000	0b0111	0b1000	0b001	ATS1CPW	Address Translate Sta 1 Current sta PL1 Write
0b1111	0b000	0b0111	0b1000	0b010	ATS1CUR	Address Translate Sta 1 Current sta Unprivileged Read
0b1111	0b000	0b0111	0b1000	0b011	ATS1CUW	Address Translate Sta 1 Current sta Unprivileged Write
0b1111	0b000	0b0111	0b1000	0b100	ATS12NSOPR	Address Translate Stages 1 and Non-secure Only PL1 Rea
0b1111	0b000	0b0111	0b1000	0b101	ATS12NSOPW	Address Translate Stages 1 and Non-secure Only PL1 Wri
0b1111	0b000	0b0111	0b1000	0b110	ATS12NSOUR	Address Translate Stages 1 and Non-secure Only Unprivileged Read
0b1111	0b000	0b0111	0b1000	0b111	ATS12NSOUW	Address Translate Stages 1 and Non-secure Only Unprivileged Write
0b1111	0b000	0b0111	0b1001	0b000	ATS1CPRP	Address Translate Sta 1 Current sta PL1 Read PAN
0b1111	0b000	0b0111	0b1001	0b001	ATS1CPWP	Address Translate Sta 1 Current sta PL1 Write PAN
0b1111	0b000	0b0111	0b1010	0b001	DCCMVAC	Data Cache li Clean by VA t PoC
0b1111	0b000	0b0111	0b1010	0b010	DCCSW	Data Cache li Clean by Set/ Way
0b1111	0b000	0b0111	0b1010	0b100	CP15DSB	Data Synchronizati Barrier System instruction

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1111	0b000	0b0111	0b1010	0b101	CP15DMB	Data Memory Barrier System instruction
0b1111	0b000	0b0111	0b1011	0b001	DCCMVAU	Data Cache Line Clean by VA to PoU
0b1111	0b000	0b0111	0b1110	0b001	DCCIMVAC	Data Cache Line Clean and Invalidate by VA to PoC
0b1111	0b000	0b0111	0b1110	0b010	DCCISW	Data Cache Line Clean and Invalidate by Set/Way
0b1111	0b000	0b1000	0b0011	0b000	TLBIALLIS	TLB Invalidate All, Inner Shareable
0b1111	0b000	0b1000	0b0011	0b001	TLBIMVAIS	TLB Invalidate by VA, Inner Shareable
0b1111	0b000	0b1000	0b0011	0b010	TLBIASIDIS	TLB Invalidate by ASID match, Inner Shareable
0b1111	0b000	0b1000	0b0011	0b011	TLBIMVAAIS	TLB Invalidate by VA, All ASID, Inner Shareable
0b1111	0b000	0b1000	0b0011	0b101	TLBIMVALIS	TLB Invalidate by VA, Last level, Inner Shareable
0b1111	0b000	0b1000	0b0011	0b111	TLBIMVAALIS	TLB Invalidate by VA, All ASID, Last level, Inner Shareable
0b1111	0b000	0b1000	0b0101	0b000	ITLBIALL	Instruction TLB Invalidate All
0b1111	0b000	0b1000	0b0101	0b001	ITLBIMVA	Instruction TLB Invalidate by VA
0b1111	0b000	0b1000	0b0101	0b010	ITLBIASID	Instruction TLB Invalidate by ASID match
0b1111	0b000	0b1000	0b0110	0b000	DTLBIALL	Data TLB Invalidate All
0b1111	0b000	0b1000	0b0110	0b001	DTLBIMVA	Data TLB Invalidate by VA
0b1111	0b000	0b1000	0b0110	0b010	DTLBIASID	Data TLB Invalidate by ASID match
0b1111	0b000	0b1000	0b0111	0b000	TLBIALL	TLB Invalidate All
0b1111	0b000	0b1000	0b0111	0b001	TLBIMVA	TLB Invalidate by VA
0b1111	0b000	0b1000	0b0111	0b010	TLBIASID	TLB Invalidate by ASID match
0b1111	0b000	0b1000	0b0111	0b011	TLBIMVAA	TLB Invalidate by VA, All ASID
0b1111	0b000	0b1000	0b0111	0b101	TLBIMVAL	TLB Invalidate by VA, Last level

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1111	0b000	0b1000	0b0111	0b111	TLBIMVAAL	TLB Invalidate by VA, All ASID Last level
0b1111	0b000	0b1001	0b1100	0b000	PMCR	Performance Monitors Control Register
0b1111	0b000	0b1001	0b1100	0b001	PMCNTENSET	Performance Monitors Count Enable Set register
0b1111	0b000	0b1001	0b1100	0b010	PMCNTENCLR	Performance Monitors Count Enable Clear register
0b1111	0b000	0b1001	0b1100	0b011	PMOVSr	Performance Monitors Overflow Flag Status Register
0b1111	0b000	0b1001	0b1100	0b100	PMSWINC	Performance Monitors Software Increment register
0b1111	0b000	0b1001	0b1100	0b101	PMSELR	Performance Monitors Event Counter Selection Register
0b1111	0b000	0b1001	0b1100	0b110	PMCEID0	Performance Monitors Common Event Identification register 0
0b1111	0b000	0b1001	0b1100	0b111	PMCEID1	Performance Monitors Common Event Identification register 1
0b1111	0b000	0b1001	0b1101	0b000	PMCCNTR	Performance Monitors Cycle Count Register
0b1111	0b000	0b1001	0b1101	0b001	PMXEVTYPER	Performance Monitors Selected Event Type Register
0b1111	0b000	0b1001	0b1101	0b010	PMXEVCNTR	Performance Monitors Selected Event Count Register
0b1111	0b000	0b1001	0b1110	0b000	PMUSERENR	Performance Monitors User Enable Register
0b1111	0b000	0b1001	0b1110	0b001	PMINTENSET	Performance Monitors Interrupt Enable Set register
0b1111	0b000	0b1001	0b1110	0b010	PMINTENCLR	Performance Monitors Interrupt Enable Clear register

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1111	0b000	0b1001	0b1110	0b011	PMOVSSET	Performance Monitors Overflow Flag Status Set register
0b1111	0b000	0b1001	0b1110	0b100	PMCEID2	Performance Monitors Common Event Identification register 2
0b1111	0b000	0b1001	0b1110	0b101	PMCEID3	Performance Monitors Common Event Identification register 3
0b1111	0b000	0b1001	0b1110	0b110	PMMIR	Performance Monitors Machine Identification Register
0b1111	0b000	0b1010	0b0011	0b000	AMAIRO	Auxiliary Memory Attribute Indirection Register 0
0b1111	0b000	0b1010	0b0011	0b001	AMAIR1	Auxiliary Memory Attribute Indirection Register 1
0b1111	0b000	0b1100	0b0000	0b000	VBAR	Vector Base Address Register
0b1111	0b000	0b1100	0b0000	0b010	RMR	Reset Management Register
0b1111	0b000	0b1100	0b0001	0b000	ISR	Interrupt Status Register
0b1111	0b000	0b1100	0b0001	0b001	DISR	Deferred Interrupt Status Register
0b1111	0b000	0b1100	0b1000	0b000	ICC_IAR0	Interrupt Controller Interrupt Acknowledge Register 0
0b1111	0b000	0b1100	0b1000	0b001	ICC_EOIR0	Interrupt Controller End Of Interrupt Register 0
0b1111	0b000	0b1100	0b1000	0b010	ICC_HPPIR0	Interrupt Controller Highest Priority Pending Interrupt Register 0
0b1111	0b000	0b1100	0b1000	0b011	ICC_BPR0	Interrupt Controller Binary Point Register 0
0b1111	0b000	0b1100	0b1000	0b1:n[1:0]	ICC_AP0R<n>	Interrupt Controller Active Priority

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
						Group 0 Registers
0b1111	0b000	0b1100	0b1001	0b0:n[1:0]	ICC_AP1R<n>	Interrupt Controller Active Priority Group 1 Registers
0b1111	0b000	0b1100	0b1011	0b001	ICC_DIR	Interrupt Controller Deactivate Interrupt Register
0b1111	0b000	0b1100	0b1011	0b011	ICC_RPR	Interrupt Controller Running Priority Register
0b1111	0b000	0b1100	0b1100	0b000	ICC_IAR1	Interrupt Controller Interrupt Acknowledge Register 1
0b1111	0b000	0b1100	0b1100	0b001	ICC_EOIR1	Interrupt Controller End Of Interrupt Register 1
0b1111	0b000	0b1100	0b1100	0b010	ICC_HPPIR1	Interrupt Controller Highest Priority Pending Interrupt Register 1
0b1111	0b000	0b1100	0b1100	0b011	ICC_BPR1	Interrupt Controller Binary Point Register 1
0b1111	0b000	0b1100	0b1100	0b100	ICC_CTLR	Interrupt Controller Control Register
0b1111	0b000	0b1100	0b1100	0b101	ICC_SRE	Interrupt Controller System Register Enable register
0b1111	0b000	0b1100	0b1100	0b110	ICC_IGRPEN0	Interrupt Controller Interrupt Group 0 Enable register
0b1111	0b000	0b1100	0b1100	0b111	ICC_IGRPEN1	Interrupt Controller Interrupt Group 1 Enable register
0b1111	0b000	0b1101	0b0000	0b000	FCSEIDR	FCSE Process ID register
0b1111	0b000	0b1101	0b0000	0b001	CONTEXTIDR	Context ID Register
0b1111	0b000	0b1101	0b0000	0b010	TPIDRURW	PL0 Read/Write Software Thread ID Register

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1111	0b000	0b1101	0b0000	0b011	TPIDRURO	PL0 Read-Only Software Thread ID Register
0b1111	0b000	0b1101	0b0000	0b100	TPIDRPRW	PL1 Software Thread ID Register
0b1111	0b000	0b1101	0b0010	0b000	AMCR	Activity Monitors Control Register
0b1111	0b000	0b1101	0b0010	0b001	AMCFGR	Activity Monitors Configuration Register
0b1111	0b000	0b1101	0b0010	0b010	AMCGCR	Activity Monitors Counter Group Configuration Register
0b1111	0b000	0b1101	0b0010	0b011	AMUSERENR	Activity Monitors User Enable Register
0b1111	0b000	0b1101	0b0010	0b100	AMCNTENCLR0	Activity Monitors Counter Enable Clear Register 0
0b1111	0b000	0b1101	0b0010	0b101	AMCNTENSET0	Activity Monitors Counter Enable Set Register 0
0b1111	0b000	0b1101	0b0011	0b000	AMCNTENCLR1	Activity Monitors Counter Enable Clear Register 1
0b1111	0b000	0b1101	0b0011	0b001	AMCNTENSET1	Activity Monitors Counter Enable Set Register 1
0b1111	0b000	0b1101	0b011:n[3]	n[2:0]	AMEVTYPER0<n>	Activity Monitors Event Type Register 0
0b1111	0b000	0b1101	0b111:n[3]	n[2:0]	AMEVTYPER1<n>	Activity Monitors Event Type Register 1
0b1111	0b000	0b1110	0b0000	0b000	CNTFRQ	Counter-timer Frequency register
0b1111	0b000	0b1110	0b0001	0b000	CNTKCTL	Counter-timer Kernel Control register
0b1111	0b000	0b1110	0b0010	0b000	CNTP_TVAL	Counter-timer Physical Time TimerValue register
0b1111	0b000	0b1110	0b0010	0b001	CNTP_CTL	Counter-timer Physical Time Control register
0b1111	0b000	0b1110	0b0011	0b000	CNTV_TVAL	Counter-timer Virtual Timer Value register

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
						TimerValue register
0b1111	0b000	0b1110	0b0011	0b001	CNTV_CTL	Counter-timer Virtual Timer Control register
0b1111	0b000	0b1110	0b10:n[4:3]	n[2:0]	PMEVCNTR<n>	Performance Monitors Event Count Register
0b1111	0b000	0b1110	0b1111	0b111	PMCCFILTR	Performance Monitors Cycle Count Filter Register
0b1111	0b000	0b1110	0b11:n[4:3]	n[2:0]	PMEVTYPER<n>	Performance Monitors Event Type Register
0b1111	0b001	0b0000	0b0000	0b000	CCSIDR	Current Cache Size ID Register
0b1111	0b001	0b0000	0b0000	0b001	CLIDR	Cache Level ID Register
0b1111	0b001	0b0000	0b0000	0b010	CCSIDR2	Current Cache Size ID Register 2
0b1111	0b001	0b0000	0b0000	0b111	AIDR	Auxiliary ID Register
0b1111	0b010	0b0000	0b0000	0b000	CSSELR	Cache Size Selection Register
0b1111	0b011	0b0100	0b0101	0b000	DSPSR	Debug Saved Program State Register
0b1111	0b011	0b0100	0b0101	0b001	DLR	Debug Link Register
0b1111	0b100	0b0000	0b0000	0b000	VPIDR	Virtualization Processor ID Register
0b1111	0b100	0b0000	0b0000	0b101	VMPIDR	Virtualization Multiprocessor ID Register
0b1111	0b100	0b0001	0b0000	0b000	HSCTLR	Hyp System Control Register
0b1111	0b100	0b0001	0b0000	0b001	HACTLR	Hyp Auxiliary Control Register
0b1111	0b100	0b0001	0b0000	0b011	HACTLR2	Hyp Auxiliary Control Register 2
0b1111	0b100	0b0001	0b0001	0b000	HCR	Hyp Configuration Register
0b1111	0b100	0b0001	0b0001	0b001	HDCR	Hyp Debug Control Register
0b1111	0b100	0b0001	0b0001	0b010	HCPTR	Hyp Architectural Feature Trap Register
0b1111	0b100	0b0001	0b0001	0b011	HSTR	Hyp System Trap Register

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1111	0b100	0b0001	0b0001	0b100	HCR2	Hyp Configuration Register 2
0b1111	0b100	0b0001	0b0001	0b111	HACR	Hyp Auxiliary Configuration Register
0b1111	0b100	0b0001	0b0010	0b001	HTRFCR	Hyp Trace Filter Control Register
0b1111	0b100	0b0010	0b0000	0b010	HTCR	Hyp Translation Control Register
0b1111	0b100	0b0010	0b0001	0b010	VTCR	Virtualization Translation Control Register
0b1111	0b100	0b0101	0b0001	0b000	HADFSR	Hyp Auxiliary Data Fault Status Register
0b1111	0b100	0b0101	0b0001	0b001	HAIFSR	Hyp Auxiliary Instruction Fault Status Register
0b1111	0b100	0b0101	0b0010	0b000	HSR	Hyp Syndrome Register
0b1111	0b100	0b0101	0b0010	0b011	VDFS	Virtual Error Exception Syndrome Register
0b1111	0b100	0b0110	0b0000	0b000	HDFAR	Hyp Data Fault Address Register
0b1111	0b100	0b0110	0b0000	0b010	HIFAR	Hyp Instruction Fault Address Register
0b1111	0b100	0b0110	0b0000	0b100	HPFAR	Hyp IPA Fault Address Register
0b1111	0b100	0b0111	0b1000	0b000	ATS1HR	Address Translate Stage 1 Hyp mode Read
0b1111	0b100	0b0111	0b1000	0b001	ATS1HW	Address Translate Stage 1 Hyp mode Write
0b1111	0b100	0b1000	0b0000	0b001	TLBIIPAS2IS	TLB Invalidate by Intermediate Physical Address, Stage 2, Inner Shareable
0b1111	0b100	0b1000	0b0000	0b101	TLBIIPAS2LIS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, Inner Shareable
0b1111	0b100	0b1000	0b0011	0b000	TLBIALLHIS	TLB Invalidate All, Hyp mode Inner Shareable

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1111	0b100	0b1000	0b0011	0b001	TLBIMVAHIS	TLB Invalidation by VA, Hyp mode, Inner Shareable
0b1111	0b100	0b1000	0b0011	0b100	TLBIALLSNHIS	TLB Invalidation All, Non-Secure Non-Hyp, Inner Shareable
0b1111	0b100	0b1000	0b0011	0b101	TLBIMVALHIS	TLB Invalidation by VA, Last level, Hyp mode, Inner Shareable
0b1111	0b100	0b1000	0b0100	0b001	TLBIIPAS2	TLB Invalidation by Intermediate Physical Address, Stage 2
0b1111	0b100	0b1000	0b0100	0b101	TLBIIPAS2L	TLB Invalidation by Intermediate Physical Address, Stage 2, Last level
0b1111	0b100	0b1000	0b0111	0b000	TLBIALLH	TLB Invalidation All, Hyp mode
0b1111	0b100	0b1000	0b0111	0b001	TLBIMVAH	TLB Invalidation by VA, Hyp mode
0b1111	0b100	0b1000	0b0111	0b100	TLBIALLSNH	TLB Invalidation All, Non-Secure Non-Hyp
0b1111	0b100	0b1000	0b0111	0b101	TLBIMVALH	TLB Invalidation by VA, Last level, Hyp mode
0b1111	0b100	0b1010	0b0010	0b000	HMAIR0	Hyp Memory Attribute Indirection Register 0
0b1111	0b100	0b1010	0b0010	0b001	HMAIR1	Hyp Memory Attribute Indirection Register 1
0b1111	0b100	0b1010	0b0011	0b000	HAMAIRO	Hyp Auxiliary Memory Attribute Indirection Register 0
0b1111	0b100	0b1010	0b0011	0b001	HAMAIR1	Hyp Auxiliary Memory Attribute Indirection Register 1
0b1111	0b100	0b1100	0b0000	0b000	HVBAR	Hyp Vector Base Address Register
0b1111	0b100	0b1100	0b0000	0b010	HRMR	Hyp Reset Management Register
0b1111	0b100	0b1100	0b0001	0b001	VDISR	Virtual Deferred Interrupt Status Register

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1111	0b100	0b1100	0b1000	0b0:n[1:0]	ICH_AP0R<n>	Interrupt Controller Hyp Active Priority Group 0 Registers
0b1111	0b100	0b1100	0b1001	0b0:n[1:0]	ICH_AP1R<n>	Interrupt Controller Hyp Active Priority Group 1 Registers
0b1111	0b100	0b1100	0b1001	0b101	ICC_HSRE	Interrupt Controller Hyp System Register Enable register
0b1111	0b100	0b1100	0b1011	0b000	ICH_HCR	Interrupt Controller Hyp Control Register
0b1111	0b100	0b1100	0b1011	0b001	ICH_VTR	Interrupt Controller VG Type Register
0b1111	0b100	0b1100	0b1011	0b010	ICH_MISR	Interrupt Controller Maintenance Interrupt Status Register
0b1111	0b100	0b1100	0b1011	0b011	ICH_EISR	Interrupt Controller Enable of Interrupt Status Register
0b1111	0b100	0b1100	0b1011	0b101	ICH_ELRSR	Interrupt Controller Empty List Register Status Register
0b1111	0b100	0b1100	0b1011	0b111	ICH_VMCR	Interrupt Controller Virtual Machine Control Register
0b1111	0b100	0b1100	0b110:n[3]	n[2:0]	ICH_LR<n>	Interrupt Controller List Registers
0b1111	0b100	0b1100	0b111:n[3]	n[2:0]	ICH_LRC<n>	Interrupt Controller List Registers
0b1111	0b100	0b1101	0b0000	0b010	HTPIDR	Hyp Software Thread ID Register
0b1111	0b100	0b1110	0b0001	0b000	CNTHCTL	Counter-timer Hyp Control register
0b1111	0b100	0b1110	0b0010	0b000	CNTHP_TVAL	Counter-timer Hyp Physical Timer TimerValue register
0b1111	0b100	0b1110	0b0010	0b001	CNTHP_CTL	Counter-timer Hyp Physical Timer Control register

coproc	opc1	Register selectors		opc2	Name	Description
		CRn	CRm			
0b1111	0b110	0b1100	0b1100	0b100	ICC_MCTLR	Interrupt Controller Monitor Control Register
0b1111	0b110	0b1100	0b1100	0b101	ICC_MSRE	Interrupt Controller Monitor System Register Enable register
0b1111	0b110	0b1100	0b1100	0b111	ICC_MGRPEN1	Interrupt Controller Monitor Interrupt Group 1 Enable register

Accessed using MCRR/MRRC:

coproc	Register selectors		Name	Description
	CRm	opc1		
0b1110	0b0001	0b0000	DBGDRAR	Debug ROM Address Register
0b1110	0b0010	0b0000	DBGDSAR	Debug Self Address Register
0b1111	0b000:n[3]	0b0:n[2:0]	AMEVCNTR0<n>	Activity Monitors Event Counter Registers 0
0b1111	0b0010	0b0000	TTBR0	Translation Table Base Register 0
0b1111	0b0010	0b0001	TTBR1	Translation Table Base Register 1
0b1111	0b0010	0b0100	HTTBR	Hyp Translation Table Base Register
0b1111	0b0010	0b0110	VTTBR	Virtualization Translation Table Base Register
0b1111	0b010:n[3]	0b0:n[2:0]	AMEVCNTR1<n>	Activity Monitors Event Counter Registers 1
0b1111	0b0111	0b0000	PAR	Physical Address Register
0b1111	0b1001	0b0000	PMCCNTR	Performance Monitors Cycle Count Register
0b1111	0b1100	0b0000	ICC_SGI1R	Interrupt Controller Software Generated Interrupt Group 1 Register
0b1111	0b1100	0b0001	ICC_ASIG1R	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
0b1111	0b1100	0b0010	ICC_SGI0R	Interrupt Controller Software Generated Interrupt Group 0 Register
0b1111	0b1110	0b0000	CNTPCT	Counter-timer Physical Count register
0b1111	0b1110	0b0001	CNTVCT	Counter-timer Virtual Count register
0b1111	0b1110	0b0010	CNTP_CVAL	Counter-timer Physical Timer CompareValue register
0b1111	0b1110	0b0011	CNTV_CVAL	Counter-timer Virtual Timer CompareValue register
0b1111	0b1110	0b0100	CNTVOFF	Counter-timer Virtual Offset register
0b1111	0b1110	0b0110	CNTHP_CVAL	Counter-timer Hyp Physical CompareValue register
0b1111	0b1110	0b1000	CNTPCTSS	Counter-timer Self-Synchronized Physical Count register
0b1111	0b1110	0b1001	CNTVCTSS	Counter-timer Self-Synchronized Virtual Count register

Accessed using MRS/MSR:

Register selectors			Name	Description
R	M	M1		
0b0	0b1	0b1110	ELR_hyp	Exception Link Register (Hyp mode)
0b1	0b0	0b1110	SPSR_fiq	Saved Program Status Register (FIQ mode)
0b1	0b1	0b0000	SPSR_irq	Saved Program Status Register (IRQ mode)
0b1	0b1	0b0010	SPSR_svc	Saved Program Status Register (Supervisor mode)
0b1	0b1	0b0100	SPSR_abt	Saved Program Status Register (Abort mode)
0b1	0b1	0b0110	SPSR_und	Saved Program Status Register (Undefined mode)
0b1	0b1	0b1100	SPSR_mon	Saved Program Status Register (Monitor mode)
0b1	0b1	0b1110	SPSR_hyp	Saved Program Status Register (Hyp mode)

Accessed using VMRS/VMSR:

Register selectors reg	Name	Description
0b0000	FPSID	Floating-Point System ID register
0b0001	FPSCR	Floating-Point Status and Control Register
0b0101	MVFR2	Media and VFP Feature Register 2
0b0110	MVFR1	Media and VFP Feature Register 1
0b0111	MVFR0	Media and VFP Feature Register 0
0b1000	FPEXC	Floating-Point Exception Control register

Registers and operations in AArch64**Accessed using AT:**

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
0b01	0b000	0b0111	0b1000	0b000	AT S1E1R	Address Translate Stage 1 EL1 Read
0b01	0b000	0b0111	0b1000	0b001	AT S1E1W	Address Translate Stage 1 EL1 Write
0b01	0b000	0b0111	0b1000	0b010	AT S1E0R	Address Translate Stage 1 EL0 Read
0b01	0b000	0b0111	0b1000	0b011	AT S1E0W	Address Translate Stage 1 EL0 Write
0b01	0b000	0b0111	0b1001	0b000	AT S1E1RP	Address Translate Stage 1 EL1 Read PAN
0b01	0b000	0b0111	0b1001	0b001	AT S1E1WP	Address Translate Stage 1 EL1 Write PAN
0b01	0b100	0b0111	0b1000	0b000	AT S1E2R	Address Translate Stage 1 EL2 Read
0b01	0b100	0b0111	0b1000	0b001	AT S1E2W	Address Translate Stage 1 EL2 Write
0b01	0b100	0b0111	0b1000	0b100	AT S12E1R	Address Translate Stages 1 and 2 EL1 Read
0b01	0b100	0b0111	0b1000	0b101	AT S12E1W	Address Translate Stages 1 and 2 EL1 Write
0b01	0b100	0b0111	0b1000	0b110	AT S12E0R	Address Translate Stages 1 and 2 EL0 Read
0b01	0b100	0b0111	0b1000	0b111	AT S12E0W	Address Translate Stages 1 and 2 EL0 Write

Register selectors				op2	Name	Description
op0	op1	CRn	CRm			
0b01	0b110	0b0111	0b1000	0b000	AT S1E3R	Address Translate Stage 1 EL3 Read
0b01	0b110	0b0111	0b1000	0b001	AT S1E3W	Address Translate Stage 1 EL3 Write

Accessed using BRB:

Register selectors				op2	Name	Description
op0	op1	CRn	CRm			
0b01	0b001	0b0111	0b0010	0b100	BRB IALL	Invalidate the Branch Record Buffer
0b01	0b001	0b0111	0b0010	0b101	BRB INJ	Branch Record Injection into the Branch Record Buffer

Accessed using CFP:

Register selectors				op2	Name	Description
op0	op1	CRn	CRm			
0b01	0b011	0b0111	0b0011	0b100	CFP RCTX	Control Flow Prediction Restriction by Context

Accessed using CPP:

Register selectors				op2	Name	Description
op0	op1	CRn	CRm			
0b01	0b011	0b0111	0b0011	0b111	CPP RCTX	Cache Prefetch Prediction Restriction by Context

Accessed using DC:

Register selectors				op2	Name	Description
op0	op1	CRn	CRm			
0b01	0b000	0b0111	0b0110	0b001	DC IVAC	Data or unified Cache line Invalidate by VA to PoC
0b01	0b000	0b0111	0b0110	0b010	DC ISW	Data or unified Cache line Invalidate by Set/Way
0b01	0b000	0b0111	0b0110	0b011	DC IGVAC	Invalidate of Allocation Tags by VA to PoC
0b01	0b000	0b0111	0b0110	0b100	DC IGSW	Invalidate of Allocation Tags by Set/Way
0b01	0b000	0b0111	0b0110	0b101	DC IGDVAC	Invalidate of Data and Allocation Tags by VA to PoC
0b01	0b000	0b0111	0b0110	0b110	DC IGDSW	Invalidate of Data and Allocation Tags by Set/Way
0b01	0b000	0b0111	0b1010	0b010	DC CSW	Data or unified Cache line Clean by Set/Way
0b01	0b000	0b0111	0b1010	0b100	DC CGSW	Clean of Allocation Tags by Set/Way
0b01	0b000	0b0111	0b1010	0b110	DC CGDSW	Clean of Data and Allocation Tags by Set/Way
0b01	0b000	0b0111	0b1110	0b010	DC CISW	Data or unified Cache line Clean and Invalidate by Set/Way
0b01	0b000	0b0111	0b1110	0b100	DC CIGSW	Clean and Invalidate of Allocation Tags by Set/Way
0b01	0b000	0b0111	0b1110	0b110	DC CIGDSW	Clean and Invalidate of Data and Allocation Tags by Set/Way

Register selectors				op2	Name	Description
op0	op1	CRn	CRm			
0b01	0b011	0b0111	0b0100	0b001	DC ZVA	Data Cache Zero by VA
0b01	0b011	0b0111	0b0100	0b011	DC GVA	Data Cache set Allocation Tag by VA
0b01	0b011	0b0111	0b0100	0b100	DC GZVA	Data Cache set Allocation Tags and Zero by VA
0b01	0b011	0b0111	0b1010	0b001	DC CVAC	Data or unified Cache line Clean by VA to PoC
0b01	0b011	0b0111	0b1010	0b011	DC CGVAC	Clean of Allocation Tags by VA to PoC
0b01	0b011	0b0111	0b1010	0b101	DC CGDVAC	Clean of Data and Allocation Tags by VA to PoC
0b01	0b011	0b0111	0b1011	0b001	DC CVAU	Data or unified Cache line Clean by VA to PoU
0b01	0b011	0b0111	0b1100	0b001	DC CVAP	Data or unified Cache line Clean by VA to PoP
0b01	0b011	0b0111	0b1100	0b011	DC CGVAP	Clean of Allocation Tags by VA to PoP
0b01	0b011	0b0111	0b1100	0b101	DC CGDVAP	Clean of Data and Allocation Tags by VA to PoP
0b01	0b011	0b0111	0b1101	0b001	DC CVADP	Data or unified Cache line Clean by VA to PoDP
0b01	0b011	0b0111	0b1101	0b011	DC CGVADP	Clean of Allocation Tags by VA to PoDP
0b01	0b011	0b0111	0b1101	0b101	DC CGDVADP	Clean of Data and Allocation Tags by VA to PoDP
0b01	0b011	0b0111	0b1110	0b001	DC CIVAC	Data or unified Cache line Clean and Invalidate by VA to PoC
0b01	0b011	0b0111	0b1110	0b011	DC CIGVAC	Clean and Invalidate of Allocation Tags by VA to PoC
0b01	0b011	0b0111	0b1110	0b101	DC CIGDVAC	Clean and Invalidate of Data and Allocation Tags by VA to PoC
0b01	0b110	0b0111	0b1110	0b001	DC CIPAPA	Data or unified Cache line Clean and Invalidate by PA to PoPA
0b01	0b110	0b0111	0b1110	0b101	DC CIGDPAPA	Clean and Invalidate of Data and Allocation Tags by PA to PoPA

Accessed using DVP:

Register selectors				op2	Name	Description
op0	op1	CRn	CRm			
0b01	0b011	0b0111	0b0011	0b101	DVP RCTX	Data Value Prediction Restriction by Context

Accessed using IC:

Register selectors				op2	Name	Description
op0	op1	CRn	CRm			
0b01	0b000	0b0111	0b0001	0b000	IC IALLUIS	Instruction Cache Invalidate All to PoU, Inner Shareable
0b01	0b000	0b0111	0b0101	0b000	IC IALLU	Instruction Cache Invalidate All to PoU
0b01	0b011	0b0111	0b0101	0b001	IC IVAU	Instruction Cache line Invalidate by VA to PoU

Accessed using MRS/MSR:

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b10	0b000	0b0000	0b0000	0b010	OSDTRRX_EL1	OS Lock Transfer Register Receive
0b10	0b000	0b0000	0b0010	0b000	MDCCINT_EL1	Monitor Interrupt Enable
0b10	0b000	0b0000	0b0010	0b010	MDSCR_EL1	Monitor System Control Register
0b10	0b000	0b0000	0b0011	0b010	OSDTRTX_EL1	OS Lock Transfer Register Transmitter
0b10	0b000	0b0000	0b0110	0b010	OSECCR_EL1	OS Lock Exception Catch Control Register
0b10	0b000	0b0000	n[3:0]	0b100	DBGBVR<n>_EL1	Debug Breakpoint Value Register
0b10	0b000	0b0000	n[3:0]	0b101	DBGBCR<n>_EL1	Debug Breakpoint Control Register
0b10	0b000	0b0000	n[3:0]	0b110	DBGWVR<n>_EL1	Debug Watchpoint Value Register
0b10	0b000	0b0000	n[3:0]	0b111	DBGWCR<n>_EL1	Debug Watchpoint Control Register
0b10	0b000	0b0001	0b0000	0b000	MDRAR_EL1	Monitor ROM Address Register
0b10	0b000	0b0001	0b0000	0b100	OSLAR_EL1	OS Lock Register
0b10	0b000	0b0001	0b0001	0b100	OSLSR_EL1	OS Lock Register
0b10	0b000	0b0001	0b0011	0b100	OSDLR_EL1	OS Double Lock Register
0b10	0b000	0b0001	0b0100	0b100	DBGPRCR_EL1	Debug Point Control Register
0b10	0b000	0b0111	0b1000	0b110	DBGCLAIMSET_EL1	Debug Claim Tag Set
0b10	0b000	0b0111	0b1001	0b110	DBGCLAIMCLR_EL1	Debug Claim Tag Clear register
0b10	0b000	0b0111	0b1110	0b110	DBGAUTHSTATUS_EL1	Debug Authentication Status register
0b10	0b001	0b0000	0b0000	0b001	TRCTRAIDR	Trace ID Register
0b10	0b001	0b0000	0b0000	0b010	TRCVICTLR	View Instruction Control Register
0b10	0b001	0b0000	0b0000	0b110	TRCIDR8	ID Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b10	0b001	0b0000	0b0000	0b111	TRCIMSPEC0	IMP DEL Register
0b10	0b001	0b0000	0b0001	0b000	TRCPRGCTLR	Program Control Register
0b10	0b001	0b0000	0b0001	0b001	TRCQCTLR	Q Element Control Register
0b10	0b001	0b0000	0b0001	0b010	TRCVIIECTLR	ViewInst Include/Control Register
0b10	0b001	0b0000	0b0001	0b110	TRCIDR9	ID Register
0b10	0b001	0b0000	0b0010	0b010	TRCVISSCTLR	ViewInst Stop Control Register
0b10	0b001	0b0000	0b0010	0b110	TRCIDR10	ID Register
0b10	0b001	0b0000	0b0011	0b000	TRCSTATR	Trace State Register
0b10	0b001	0b0000	0b0011	0b010	TRCVIPCSSCTLR	ViewInst Stop PE Compare Control Register
0b10	0b001	0b0000	0b0011	0b110	TRCIDR11	ID Register
0b10	0b001	0b0000	0b00:n[1:0]	0b100	TRCSEQEVR<n>	Sequence State Trace Control Register
0b10	0b001	0b0000	0b00:n[1:0]	0b101	TRCCNTRLDVR<n>	Counter Value Register <n>
0b10	0b001	0b0000	0b0100	0b000	TRCCONFIGR	Trace Configuration Register
0b10	0b001	0b0000	0b0100	0b110	TRCIDR12	ID Register
0b10	0b001	0b0000	0b0101	0b110	TRCIDR13	ID Register
0b10	0b001	0b0000	0b0110	0b000	TRCAUXCTLR	Auxiliary Control Register
0b10	0b001	0b0000	0b0110	0b100	TRCSEQRSTEVR	Sequence Reset Control Register
0b10	0b001	0b0000	0b0111	0b100	TRCSEQSTR	Sequence State Register
0b10	0b001	0b0000	0b01:n[1:0]	0b101	TRCCNTCTLR<n>	Counter Control Register
0b10	0b001	0b0000	0b0:n[2:0]	0b111	TRCIMSPEC<n>	IMP DEL Register
0b10	0b001	0b0000	0b1000	0b000	TRCEVENTCTL0R	Event Control Register
0b10	0b001	0b0000	0b1000	0b111	TRCIDR0	ID Register
0b10	0b001	0b0000	0b1001	0b000	TRCEVENTCTL1R	Event Control Register
0b10	0b001	0b0000	0b1001	0b111	TRCIDR1	ID Register
0b10	0b001	0b0000	0b1010	0b000	TRCRSR	Resource Status Register
0b10	0b001	0b0000	0b1010	0b111	TRCIDR2	ID Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b10	0b001	0b0000	0b1011	0b000	TRCSTALLCTLR	Stall Control Register
0b10	0b001	0b0000	0b1011	0b111	TRCIDR3	ID Register
0b10	0b001	0b0000	0b10:n[1:0]	0b100	TRCEXTINSEL<n>	External Select Register <n>
0b10	0b001	0b0000	0b10:n[1:0]	0b101	TRCCNTVR<n>	Counter Register <n>
0b10	0b001	0b0000	0b1100	0b000	TRCTSCTLR	Timestamp Control Register
0b10	0b001	0b0000	0b1100	0b111	TRCIDR4	ID Register
0b10	0b001	0b0000	0b1101	0b000	TRCSYNCPR	Synchronous Period Register
0b10	0b001	0b0000	0b1101	0b111	TRCIDR5	ID Register
0b10	0b001	0b0000	0b1110	0b000	TRCCCCTLR	Cycle Control Register
0b10	0b001	0b0000	0b1110	0b111	TRCIDR6	ID Register
0b10	0b001	0b0000	0b1111	0b000	TRCBBCTLR	Branch Broadcast Control Register
0b10	0b001	0b0000	0b1111	0b111	TRCIDR7	ID Register
0b10	0b001	0b0001	0b0001	0b100	TRCOSLSR	Trace OS Status Register
0b10	0b001	0b0001	0b0:n[2:0]	0b010	TRCSSCCR<n>	Single-shot Comparator Control Register <n>
0b10	0b001	0b0001	0b0:n[2:0]	0b011	TRCSSPCICR<n>	Single-shot Processing Element Comparator Input Control Register <n>
0b10	0b001	0b0001	0b1:n[2:0]	0b010	TRCSSCSR<n>	Single-shot Comparator Control Register <n>
0b10	0b001	0b0001	n[3:0]	0b00:n[4]	TRCRSCTLR<n>	Resource Selection Control Register <n>
0b10	0b001	0b0010	n[2:0]:0b0	0b00:n[3]	TRCACVR<n>	Address Comparator Value Register <n>
0b10	0b001	0b0010	n[2:0]:0b0	0b01:n[3]	TRCACATR<n>	Address Comparator Access Threshold Register <n>
0b10	0b001	0b0011	0b0000	0b010	TRCCIDCCTLR0	Context Identifier Comparator Control Register 0
0b10	0b001	0b0011	0b0001	0b010	TRCCIDCCTLR1	Context Identifier Comparator Control Register 1

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
						Control Register
0b10	0b001	0b0011	0b0010	0b010	TRCVMIDCCTLR0	Virtual C Identifier Comparison Control Register
0b10	0b001	0b0011	0b0011	0b010	TRCVMIDCCTLR1	Virtual C Identifier Comparison Control Register
0b10	0b001	0b0011	n[2:0]:0b0	0b000	TRCCIDCVR<n>	Context Identifier Comparison Value Register <n>
0b10	0b001	0b0011	n[2:0]:0b0	0b001	TRCVMIDCVR<n>	Virtual C Identifier Comparison Value Register <n>
0b10	0b001	0b0111	0b0010	0b111	TRCDEVID	Device Configuration Register
0b10	0b001	0b0111	0b1000	0b110	TRCCLAIMSET	Claim Tag Register
0b10	0b001	0b0111	0b1001	0b110	TRCCLAIMCLR	Claim Tag Register
0b10	0b001	0b0111	0b1110	0b110	TRCAUTHSTATUS	Authentication Status Register
0b10	0b001	0b0111	0b1111	0b110	TRCDEVARCH	Device Architecture Register
0b10	0b001	0b1000	n[3:0]	n[4]:0b00	BRBINF<n>_EL1	Branch Instruction Buffer Information Register
0b10	0b001	0b1000	n[3:0]	n[4]:0b01	BRBSRC<n>_EL1	Branch Instruction Buffer Source Address Register
0b10	0b001	0b1000	n[3:0]	n[4]:0b10	BRBTGT<n>_EL1	Branch Instruction Buffer Target Address Register
0b10	0b001	0b1001	0b0000	0b000	BRBCR_EL1	Branch Instruction Buffer Control Register
0b10	0b001	0b1001	0b0000	0b001	BRBFCR_EL1	Branch Instruction Buffer Flush Control Register
0b10	0b001	0b1001	0b0000	0b010	BRBTS_EL1	Branch Instruction Buffer Timestamp Register
0b10	0b001	0b1001	0b0001	0b000	BRBINFINJ_EL1	Branch Instruction Buffer Information Injection Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b10	0b001	0b1001	0b0001	0b001	BRBSRCINJ_EL1	Branch Instruction Buffer Source Address Injection Register
0b10	0b001	0b1001	0b0001	0b010	BRBTGTINJ_EL1	Branch Instruction Buffer Target Address Injection Register
0b10	0b001	0b1001	0b0010	0b000	BRBIDR0_EL1	Branch Instruction Buffer ID Register
0b10	0b011	0b0000	0b0001	0b000	MDCCSR_EL0	Monitor Status Register
0b10	0b011	0b0000	0b0100	0b000	DBGDTR_EL0	Debug Data Transfer Register duplex
0b10	0b011	0b0000	0b0101	0b000	DBGDTRRX_EL0	Debug Data Transfer Register Receive
0b10	0b011	0b0000	0b0101	0b000	DBGDTRTX_EL0	Debug Data Transfer Register Transmitter
0b10	0b100	0b0000	0b0111	0b000	DBGVCR32_EL2	Debug Viewpoint Catch Register
0b10	0b100	0b1001	0b0000	0b000	BRBCR_EL2	Branch Instruction Buffer Catch Register
0b11	0b000	0b0000	0b0000	0b000	MIDR_EL1	Main ID Register
0b11	0b000	0b0000	0b0000	0b101	MPIDR_EL1	Multiprocessor Affinity Register
0b11	0b000	0b0000	0b0000	0b110	REVIDR_EL1	Revision Register
0b11	0b000	0b0000	0b0001	0b000	ID_PFR0_EL1	AArch32 Processor Feature Register
0b11	0b000	0b0000	0b0001	0b001	ID_PFR1_EL1	AArch32 Processor Feature Register
0b11	0b000	0b0000	0b0001	0b010	ID_DFR0_EL1	AArch32 Feature Register
0b11	0b000	0b0000	0b0001	0b011	ID_AFR0_EL1	AArch32 Auxiliary Feature Register
0b11	0b000	0b0000	0b0001	0b100	ID_MMFR0_EL1	AArch32 Memory Feature Register
0b11	0b000	0b0000	0b0001	0b101	ID_MMFR1_EL1	AArch32 Memory Feature Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b000	0b0000	0b0001	0b110	ID_MMFR2_EL1	AArch32 Memory Feature Register
0b11	0b000	0b0000	0b0001	0b111	ID_MMFR3_EL1	AArch32 Memory Feature Register
0b11	0b000	0b0000	0b0010	0b000	ID_ISAR0_EL1	AArch32 Instruction Attribute Register
0b11	0b000	0b0000	0b0010	0b001	ID_ISAR1_EL1	AArch32 Instruction Attribute Register
0b11	0b000	0b0000	0b0010	0b010	ID_ISAR2_EL1	AArch32 Instruction Attribute Register
0b11	0b000	0b0000	0b0010	0b011	ID_ISAR3_EL1	AArch32 Instruction Attribute Register
0b11	0b000	0b0000	0b0010	0b100	ID_ISAR4_EL1	AArch32 Instruction Attribute Register
0b11	0b000	0b0000	0b0010	0b101	ID_ISAR5_EL1	AArch32 Instruction Attribute Register
0b11	0b000	0b0000	0b0010	0b110	ID_MMFR4_EL1	AArch32 Memory Feature Register
0b11	0b000	0b0000	0b0010	0b111	ID_ISAR6_EL1	AArch32 Instruction Attribute Register
0b11	0b000	0b0000	0b0011	0b000	MVFR0_EL1	AArch32 and VFP Feature Register
0b11	0b000	0b0000	0b0011	0b001	MVFR1_EL1	AArch32 and VFP Feature Register
0b11	0b000	0b0000	0b0011	0b010	MVFR2_EL1	AArch32 and VFP Feature Register
0b11	0b000	0b0000	0b0011	0b100	ID_PFR2_EL1	AArch32 Processor Feature Register
0b11	0b000	0b0000	0b0011	0b101	ID_DFR1_EL1	Debug Feature Register
0b11	0b000	0b0000	0b0011	0b110	ID_MMFR5_EL1	AArch32 Memory Feature Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b000	0b0000	0b0100	0b000	ID_AA64PFR0_EL1	AArch64 Process Feature Register
0b11	0b000	0b0000	0b0100	0b001	ID_AA64PFR1_EL1	AArch64 Process Feature Register
0b11	0b000	0b0000	0b0100	0b100	ID_AA64ZFR0_EL1	SVE Feature register
0b11	0b000	0b0000	0b0100	0b101	ID_AA64SMFR0_EL1	SME Feature register
0b11	0b000	0b0000	0b0101	0b000	ID_AA64DFR0_EL1	AArch64 Feature Register
0b11	0b000	0b0000	0b0101	0b001	ID_AA64DFR1_EL1	AArch64 Feature Register
0b11	0b000	0b0000	0b0101	0b100	ID_AA64AFR0_EL1	AArch64 Auxiliary Feature Register
0b11	0b000	0b0000	0b0101	0b101	ID_AA64AFR1_EL1	AArch64 Auxiliary Feature Register
0b11	0b000	0b0000	0b0110	0b000	ID_AA64ISAR0_EL1	AArch64 Instruction Attribute Register
0b11	0b000	0b0000	0b0110	0b001	ID_AA64ISAR1_EL1	AArch64 Instruction Attribute Register
0b11	0b000	0b0000	0b0110	0b010	ID_AA64ISAR2_EL1	AArch64 Instruction Attribute Register
0b11	0b000	0b0000	0b0111	0b000	ID_AA64MMFR0_EL1	AArch64 Memory Feature Register
0b11	0b000	0b0000	0b0111	0b001	ID_AA64MMFR1_EL1	AArch64 Memory Feature Register
0b11	0b000	0b0000	0b0111	0b010	ID_AA64MMFR2_EL1	AArch64 Memory Feature Register
0b11	0b000	0b0001	0b0000	0b000	SCTLR_EL1	System Control Register
0b11	0b000	0b0001	0b0000	0b001	ACTLR_EL1	Auxiliary Control Register
0b11	0b000	0b0001	0b0000	0b010	CPACR_EL1	Architectural Feature Control Register
0b11	0b000	0b0001	0b0000	0b101	RGSR_EL1	Random Allocation Seed Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b000	0b0001	0b0000	0b110	GCR_EL1	Tag Control Register
0b11	0b000	0b0001	0b0010	0b000	ZCR_EL1	SVE Control Register
0b11	0b000	0b0001	0b0010	0b001	TRFCR_EL1	Trace Filter Control Register
0b11	0b000	0b0001	0b0010	0b100	SMPRI_EL1	Streaming Mode Priority Register
0b11	0b000	0b0001	0b0010	0b110	SMCR_EL1	SME Control Register
0b11	0b000	0b0010	0b0000	0b000	TTBR0_EL1	Translation Table Base Register (EL1)
0b11	0b000	0b0010	0b0000	0b001	TTBR1_EL1	Translation Table Base Register (EL1)
0b11	0b000	0b0010	0b0000	0b010	TCR_EL1	Translation Control Register
0b11	0b000	0b0010	0b0001	0b000	APIAKeyLo_EL1	Pointer Authentication Key A for Instructions (bits[63:0])
0b11	0b000	0b0010	0b0001	0b001	APIAKeyHi_EL1	Pointer Authentication Key A for Instructions (bits[127:64])
0b11	0b000	0b0010	0b0001	0b010	APIBKeyLo_EL1	Pointer Authentication Key B for Instructions (bits[63:0])
0b11	0b000	0b0010	0b0001	0b011	APIBKeyHi_EL1	Pointer Authentication Key B for Instructions (bits[127:64])
0b11	0b000	0b0010	0b0010	0b000	APDAKeyLo_EL1	Pointer Authentication Key A for Data (bits[63:0])
0b11	0b000	0b0010	0b0010	0b001	APDAKeyHi_EL1	Pointer Authentication Key A for Data (bits[127:64])
0b11	0b000	0b0010	0b0010	0b010	APDBKeyLo_EL1	Pointer Authentication Key B for Data (bits[63:0])
0b11	0b000	0b0010	0b0010	0b011	APDBKeyHi_EL1	Pointer Authentication Key B for Data (bits[127:64])
0b11	0b000	0b0010	0b0011	0b000	APGAKeyLo_EL1	Pointer Authentication Key A for Global Asynchronous Debug (bits[63:0])

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b000	0b0010	0b0011	0b001	APGAKeyHi_EL1	Key A for Authentication (bits[63:0])
0b11	0b000	0b0100	0b0000	0b000	SPSR_EL1	Saved Program Status Register (EL1)
0b11	0b000	0b0100	0b0000	0b001	ELR_EL1	Exception Link Register
0b11	0b000	0b0100	0b0001	0b000	SP_EL0	Stack Pointer (EL0)
0b11	0b000	0b0100	0b0010	0b000	SPSel	Stack Pointer Select
0b00	0b000	0b0100	-	0b101	SPSel	Stack Pointer Select
0b11	0b000	0b0100	0b0010	0b010	CurrentEL	Current Exception Link
0b11	0b000	0b0100	0b0010	0b011	PAN	Privileged Access Not Allowed
0b00	0b000	0b0100	-	0b100	PAN	Privileged Access Not Allowed
0b11	0b000	0b0100	0b0010	0b100	UAO	User Access Override
0b00	0b000	0b0100	-	0b011	UAO	User Access Override
0b11	0b000	0b0100	0b0011	0b000	ALLINT	All Interrupt Mask Bits
0b00	0b001	0b0100	-	0b000	ALLINT	All Interrupt Mask Bits
0b11	0b000	0b0100	0b0110	0b000	ICC_PMR_EL1	Interrupt Controller Priority Register
0b11	0b000	0b0101	0b0001	0b000	AFSR0_EL1	Auxiliary Status Register 0 (EL1)
0b11	0b000	0b0101	0b0001	0b001	AFSR1_EL1	Auxiliary Status Register 1 (EL1)
0b11	0b000	0b0101	0b0010	0b000	ESR_EL1	Exception Syndrome Register
0b11	0b000	0b0101	0b0011	0b000	ERRIDR_EL1	Error Register
0b11	0b000	0b0101	0b0011	0b001	ERRSELR_EL1	Error Register Select
0b11	0b000	0b0101	0b0100	0b000	ERXFR_EL1	Selected Record Index Register
0b11	0b000	0b0101	0b0100	0b001	ERXCTLR_EL1	Selected Record Control Register
0b11	0b000	0b0101	0b0100	0b010	ERXSTATUS_EL1	Selected Record Index Status Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b000	0b0101	0b0100	0b011	ERXADDR_EL1	Selected Record Address Register
0b11	0b000	0b0101	0b0100	0b100	ERXPFGF_EL1	Selected Pseudo-fault Generation Feature register
0b11	0b000	0b0101	0b0100	0b101	ERXPFGCTL_EL1	Selected Pseudo-fault Generation Control register
0b11	0b000	0b0101	0b0100	0b110	ERXPFGCDN_EL1	Selected Pseudo-fault Generation Countdown register
0b11	0b000	0b0101	0b0101	0b000	ERXMISC0_EL1	Selected Record Miscellaneous Register
0b11	0b000	0b0101	0b0101	0b001	ERXMISC1_EL1	Selected Record Miscellaneous Register
0b11	0b000	0b0101	0b0101	0b010	ERXMISC2_EL1	Selected Record Miscellaneous Register
0b11	0b000	0b0101	0b0101	0b011	ERXMISC3_EL1	Selected Record Miscellaneous Register
0b11	0b000	0b0101	0b0110	0b000	TFSR_EL1	Tag Fault Status Register (EL1)
0b11	0b000	0b0101	0b0110	0b001	TFSRE0_EL1	Tag Fault Status Register (EL0).
0b11	0b000	0b0110	0b0000	0b000	FAR_EL1	Fault Address Register
0b11	0b000	0b0111	0b0100	0b000	PAR_EL1	Physical Address Register
0b11	0b000	0b1001	0b1001	0b000	PMSCR_EL1	Statistic Profiling Control Register
0b11	0b000	0b1001	0b1001	0b001	PMSNEVFR_EL1	Sampling Inverted Filter Register
0b11	0b000	0b1001	0b1001	0b010	PMSICR_EL1	Sampling Interval Counter Register
0b11	0b000	0b1001	0b1001	0b011	PMSIRR_EL1	Sampling Interval Register
0b11	0b000	0b1001	0b1001	0b100	PMSFCR_EL1	Sampling Control Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b000	0b1001	0b1001	0b101	PMSEVFR_EL1	Sampling Filter Register
0b11	0b000	0b1001	0b1001	0b110	PMSLATFR_EL1	Sampling Latency Register
0b11	0b000	0b1001	0b1001	0b111	PMSIDR_EL1	Sampling Profiling Register
0b11	0b000	0b1001	0b1010	0b000	PMBLIMITR_EL1	Profiling Limit Address Register
0b11	0b000	0b1001	0b1010	0b001	PMBPTR_EL1	Profiling Write Pointer Register
0b11	0b000	0b1001	0b1010	0b011	PMBSR_EL1	Profiling Status/syndrom Register
0b11	0b000	0b1001	0b1010	0b111	PMBIDR_EL1	Profiling ID Register
0b11	0b000	0b1001	0b1011	0b000	TRBLIMITR_EL1	Trace Buffer Limit Address Register
0b11	0b000	0b1001	0b1011	0b001	TRBPTR_EL1	Trace Buffer Write Pointer Register
0b11	0b000	0b1001	0b1011	0b010	TRBBASER_EL1	Trace Buffer Base Address Register
0b11	0b000	0b1001	0b1011	0b011	TRBSR_EL1	Trace Buffer Status/syndrom Register
0b11	0b000	0b1001	0b1011	0b100	TRBMAR_EL1	Trace Buffer Memory Attribute Register
0b11	0b000	0b1001	0b1011	0b110	TRBTRG_EL1	Trace Buffer Trigger Counter Register
0b11	0b000	0b1001	0b1011	0b111	TRBIDR_EL1	Trace Buffer Register
0b11	0b000	0b1001	0b1110	0b001	PMINTENSET_EL1	Performance Monitor Interrupt Enable Set register
0b11	0b000	0b1001	0b1110	0b010	PMINTENCLR_EL1	Performance Monitor Interrupt Enable Clear register
0b11	0b000	0b1001	0b1110	0b110	PMMIR_EL1	Performance Monitor Machine Identification Register
0b11	0b000	0b1010	0b0010	0b000	MAIR_EL1	Memory Attribute Indirect Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b000	0b1010	0b0011	0b000	AMAIR_EL1	Auxiliary Memory Attribute Indirection Register
0b11	0b000	0b1010	0b0100	0b000	LORSA_EL1	LORegion Address
0b11	0b000	0b1010	0b0100	0b001	LOREA_EL1	LORegion Address
0b11	0b000	0b1010	0b0100	0b010	LORN_EL1	LORegion Number
0b11	0b000	0b1010	0b0100	0b011	LORC_EL1	LORegion Control
0b11	0b000	0b1010	0b0100	0b100	MPAMIDR_EL1	MPAM ID Register
0b11	0b000	0b1010	0b0100	0b111	LORID_EL1	LORegion (EL1)
0b11	0b000	0b1010	0b0101	0b000	MPAM1_EL1	MPAM1 Register
0b11	0b000	0b1010	0b0101	0b001	MPAM0_EL1	MPAM0 Register
0b11	0b000	0b1010	0b0101	0b011	MPAMSM_EL1	MPAM Streaming Mode Register
0b11	0b000	0b1100	0b0000	0b000	VBAR_EL1	Vector Base Address Register
0b11	0b000	0b1100	0b0000	0b001	RVBAR_EL1	Reset Vector Base Address Register and EL3 implementation
0b11	0b000	0b1100	0b0000	0b010	RMR_EL1	Reset Management Register
0b11	0b000	0b1100	0b0001	0b000	ISR_EL1	Interrupt Status Register
0b11	0b000	0b1100	0b0001	0b001	DISR_EL1	Deferred Interrupt Status Register
0b11	0b000	0b1100	0b1000	0b000	ICC_IAR0_EL1	Interrupt Controller Interrupt Acknowledgment Register
0b11	0b000	0b1100	0b1000	0b001	ICC_EOIR0_EL1	Interrupt Controller Of Interrupt Register
0b11	0b000	0b1100	0b1000	0b010	ICC_HPPIR0_EL1	Interrupt Controller Highest Pending Interrupt Register
0b11	0b000	0b1100	0b1000	0b011	ICC_BPR0_EL1	Interrupt Controller Binary Priority Register
0b11	0b000	0b1100	0b1000	0b1:n[1:0]	ICC_AP0R<n>_EL1	Interrupt Controller

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
						Active P Group 0 Register
0b11	0b000	0b1100	0b1001	0b0:n[1:0]	ICC_AP1R<n>_EL1	Interrupt Controlle Active P Group 1 Register
0b11	0b000	0b1100	0b1001	0b101	ICC_NMIAR1_EL1	Interrupt Controlle maskabl Interrupt Acknowl Register
0b11	0b000	0b1100	0b1011	0b001	ICC_DIR_EL1	Interrupt Controlle Deactiva Interrupt Register
0b11	0b000	0b1100	0b1011	0b011	ICC_RPR_EL1	Interrupt Controlle Running Priority Register
0b11	0b000	0b1100	0b1011	0b101	ICC_SGI1R_EL1	Interrupt Controlle Software Generato Interrupt 1 Regist
0b11	0b000	0b1100	0b1011	0b110	ICC_ASGI1R_EL1	Interrupt Controlle Software Generato Interrupt 1 Regist
0b11	0b000	0b1100	0b1011	0b111	ICC_SGI0R_EL1	Interrupt Controlle Software Generato Interrupt 0 Regist
0b11	0b000	0b1100	0b1100	0b000	ICC_IAR1_EL1	Interrupt Controlle Interrupt Acknowl Register
0b11	0b000	0b1100	0b1100	0b001	ICC_EOIR1_EL1	Interrupt Controlle Of Intern Register
0b11	0b000	0b1100	0b1100	0b010	ICC_HPPIR1_EL1	Interrupt Controlle Highest Pending Interrupt Register
0b11	0b000	0b1100	0b1100	0b011	ICC_BPR1_EL1	Interrupt Controlle Binary P Register
0b11	0b000	0b1100	0b1100	0b100	ICC_CTLR_EL1	Interrupt Controlle

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
						Control Register
0b11	0b000	0b1100	0b1100	0b101	ICC_SRE_EL1	Interrupt Controller System Register
0b11	0b000	0b1100	0b1100	0b110	ICC_IGRPEN0_EL1	Interrupt Controller Interrupt 0 Enable register
0b11	0b000	0b1100	0b1100	0b111	ICC_IGRPEN1_EL1	Interrupt Controller Interrupt 1 Enable register
0b11	0b000	0b1101	0b0000	0b001	CONTEXTIDR_EL1	Context Register
0b11	0b000	0b1101	0b0000	0b100	TPIDR_EL1	EL1 Software Thread ID Register
0b11	0b000	0b1101	0b0000	0b101	ACCDATA_EL1	Accelerator Data
0b11	0b000	0b1101	0b0000	0b111	SCXTNUM_EL1	EL1 Real-time Software Context Number
0b11	0b000	0b1110	0b0001	0b000	CNTKCTL_EL1	Counter Kernel Control register
0b11	0b001	0b0000	0b0000	0b000	CCSIDR_EL1	Current Size ID Register
0b11	0b001	0b0000	0b0000	0b001	CLIDR_EL1	Cache Level Register
0b11	0b001	0b0000	0b0000	0b010	CCSIDR2_EL1	Current Size ID Register
0b11	0b001	0b0000	0b0000	0b100	GMID_EL1	Multiple transfer register
0b11	0b001	0b0000	0b0000	0b110	SMIDR_EL1	Streaming Mode Identifier Register
0b11	0b001	0b0000	0b0000	0b111	AIDR_EL1	Auxiliary Register
0b11	0b010	0b0000	0b0000	0b000	CSSELR_EL1	Cache Selection Register
0b11	0b011	0b0000	0b0000	0b001	CTR_EL0	Cache Tag Register
0b11	0b011	0b0000	0b0000	0b111	DCZID_EL0	Data Cache Zero ID
0b11	0b011	0b0010	0b0100	0b000	RNDR	Random Number
0b11	0b011	0b0010	0b0100	0b001	RNDRRS	Reseeded Random Number
0b11	0b011	0b0100	0b0010	0b000	NZCV	Condition

		Register selectors		op2	Name	Description
op0	op1	CRn	CRm			
0b11	0b011	0b0100	0b0010	0b001	DAIF	Interrupt Bits
0b11	0b011	0b0100	0b0010	0b010	SVCR	Streaming Vector C Register
0b11	0b011	0b0100	0b0010	0b101	DIT	Data Independ Timing
0b00	0b011	0b0100	-	0b010	DIT	Data Independ Timing
0b11	0b011	0b0100	0b0010	0b110	SSBS	Speculat Store By Safe
0b00	0b011	0b0100	-	0b001	SSBS	Speculat Store By Safe
0b11	0b011	0b0100	0b0010	0b111	TCO	Tag Che Override
0b00	0b011	0b0100	-	0b100	TCO	Tag Che Override
0b11	0b011	0b0100	0b0100	0b000	FPCR	Floating Control Register
0b11	0b011	0b0100	0b0100	0b001	FPSR	Floating Status R
0b11	0b011	0b0100	0b0101	0b000	DSPSR_ELO	Debug S Program Register
0b11	0b011	0b0100	0b0101	0b001	DLR_ELO	Debug L Register
0b11	0b011	0b1001	0b1100	0b000	PMCR_ELO	Perform Monitor Control Register
0b11	0b011	0b1001	0b1100	0b001	PMCNTENSET_ELO	Perform Monitor Enable S register
0b11	0b011	0b1001	0b1100	0b010	PMCNTENCLR_ELO	Perform Monitor Enable C register
0b11	0b011	0b1001	0b1100	0b011	PMOVSCLR_ELO	Perform Monitor Overflow Status C Register
0b11	0b011	0b1001	0b1100	0b100	PMSWINC_ELO	Perform Monitor Software Increme register
0b11	0b011	0b1001	0b1100	0b101	PMSELR_ELO	Perform Monitor Counter Selection Register
0b11	0b011	0b1001	0b1100	0b110	PMCEID0_ELO	Perform Monitor Common

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
						Identific register
0b11	0b011	0b1001	0b1100	0b111	PMCEID1_ELO	Performa Monitor: Common Identific register
0b11	0b011	0b1001	0b1101	0b000	PMCCNTR_ELO	Performa Monitor: Count R
0b11	0b011	0b1001	0b1101	0b001	PMXEVTYPER_ELO	Performa Monitor: Selected Type Re
0b11	0b011	0b1001	0b1101	0b010	PMXEVCNTR_ELO	Performa Monitor: Selected Count R
0b11	0b011	0b1001	0b1110	0b000	PMUSERENR_ELO	Performa Monitor: Enable F
0b11	0b011	0b1001	0b1110	0b011	PMOVSSET_ELO	Performa Monitor: Overflow Status S register
0b11	0b011	0b1101	0b0000	0b010	TPIDR_ELO	EL0 Rea Software Thread I Register
0b11	0b011	0b1101	0b0000	0b011	TPIDRRO_ELO	EL0 Rea Software Thread I Register
0b11	0b011	0b1101	0b0000	0b101	TPIDR2_ELO	EL0 Rea Software Thread I Register
0b11	0b011	0b1101	0b0000	0b111	SCXTNUM_ELO	EL0 Rea Software Context Number
0b11	0b011	0b1101	0b0010	0b000	AMCR_ELO	Activity Monitor: Control Register
0b11	0b011	0b1101	0b0010	0b001	AMCFGR_ELO	Activity Monitor: Configur Register
0b11	0b011	0b1101	0b0010	0b010	AMCGCR_ELO	Activity Monitor: Counter Configur Register
0b11	0b011	0b1101	0b0010	0b011	AMUSERENR_ELO	Activity Monitor: Enable F
0b11	0b011	0b1101	0b0010	0b100	AMCNTENCLR0_ELO	Activity Monitor: Enable C Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b011	0b1101	0b0010	0b101	AMCNTENSET0_EL0	Activity Monitor: Enable S Register
0b11	0b011	0b1101	0b0010	0b110	AMCG1IDR_EL0	Activity Monitor: Counter 1 Identif Register
0b11	0b011	0b1101	0b0011	0b000	AMCNTENCLR1_EL0	Activity Monitor: Enable C Register
0b11	0b011	0b1101	0b0011	0b001	AMCNTENSET1_EL0	Activity Monitor: Enable S Register
0b11	0b011	0b1101	0b010:n[3]	n[2:0]	AMEVCNTR0<n>_EL0	Activity Monitor: Counter Register
0b11	0b011	0b1101	0b011:n[3]	n[2:0]	AMEVTYPER0<n>_EL0	Activity Monitor: Type Re 0
0b11	0b011	0b1101	0b110:n[3]	n[2:0]	AMEVCNTR1<n>_EL0	Activity Monitor: Counter Register
0b11	0b011	0b1101	0b111:n[3]	n[2:0]	AMEVTYPER1<n>_EL0	Activity Monitor: Type Re 1
0b11	0b011	0b1110	0b0000	0b000	CNTFRQ_EL0	Counter: Frequen register
0b11	0b011	0b1110	0b0000	0b001	CNTPCT_EL0	Counter: Physical register
0b11	0b011	0b1110	0b0000	0b010	CNTVCT_EL0	Counter: Virtual C register
0b11	0b011	0b1110	0b0000	0b101	CNTPCTSS_EL0	Counter: Self-Synchro Physical register
0b11	0b011	0b1110	0b0000	0b110	CNTVCTSS_EL0	Counter: Self-Synchro Virtual C register
0b11	0b011	0b1110	0b0010	0b000	CNTP_TVAL_EL0	Counter: Physical TimerVa register
0b11	0b011	0b1110	0b0010	0b001	CNTP_CTL_EL0	Counter: Physical Control
0b11	0b011	0b1110	0b0010	0b010	CNTP_CVAL_EL0	Counter: Physical Compare register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b011	0b1110	0b0011	0b000	CNTV_TVAL_EL0	Counter Virtual Timer Value register
0b11	0b011	0b1110	0b0011	0b001	CNTV_CTL_EL0	Counter Virtual Timer Control register
0b11	0b011	0b1110	0b0011	0b010	CNTV_CVAL_EL0	Counter Virtual Timer Compare register
0b11	0b011	0b1110	0b10:n[4:3]	n[2:0]	PMEVCNTR<n>_EL0	Performance Monitor Count Register
0b11	0b011	0b1110	0b1111	0b111	PMCCFILTR_EL0	Performance Monitor Count Filter Register
0b11	0b011	0b1110	0b11:n[4:3]	n[2:0]	PMEVTYPER<n>_EL0	Performance Monitor Type Register
0b11	0b100	0b0000	0b0000	0b000	VPIDR_EL2	Virtualization Process ID Register
0b11	0b100	0b0000	0b0000	0b101	VMPIDR_EL2	Virtualization Multiprocessor ID Register
0b11	0b100	0b0001	0b0000	0b000	SCTLR_EL2	System Control Register
0b11	0b100	0b0001	0b0000	0b001	ACTLR_EL2	Auxiliary Control Register
0b11	0b100	0b0001	0b0001	0b000	HCR_EL2	Hypervisor Configuration Register
0b11	0b100	0b0001	0b0001	0b001	MDCR_EL2	Monitor Configuration Register
0b11	0b100	0b0001	0b0001	0b010	CPTR_EL2	Architectural Feature Register
0b11	0b100	0b0001	0b0001	0b011	HSTR_EL2	Hypervisor System Trap Register
0b11	0b100	0b0001	0b0001	0b100	HFGTR_EL2	Hypervisor Fine-Grained Read Trap Register
0b11	0b100	0b0001	0b0001	0b101	HFGWTR_EL2	Hypervisor Fine-Grained Write Trap Register
0b11	0b100	0b0001	0b0001	0b110	HFGITR_EL2	Hypervisor Fine-Grained Instruction Trap Register
0b11	0b100	0b0001	0b0001	0b111	HACR_EL2	Hypervisor Auxiliary Control Register
0b11	0b100	0b0001	0b0010	0b000	ZCR_EL2	SVE Control Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b100	0b0001	0b0010	0b001	TRFCR_EL2	Trace Filter Control Register
0b11	0b100	0b0001	0b0010	0b010	HCRX_EL2	Extended Hypervisor Configuration Register
0b11	0b100	0b0001	0b0010	0b101	SMPRIMAP_EL2	Streaming Mode Privilege Mapping Register
0b11	0b100	0b0001	0b0010	0b110	SMCR_EL2	SME Control Register
0b11	0b100	0b0001	0b0011	0b001	SDER32_EL2	AArch32 Secure EL2 Enable Flag
0b11	0b100	0b0010	0b0000	0b000	TTBR0_EL2	Translation Table Base Register (EL2)
0b11	0b100	0b0010	0b0000	0b001	TTBR1_EL2	Translation Table Base Register (EL2)
0b11	0b100	0b0010	0b0000	0b010	TCR_EL2	Translation Control Register
0b11	0b100	0b0010	0b0001	0b000	VTTBR_EL2	Virtualization Translation Table Base Register
0b11	0b100	0b0010	0b0001	0b010	VTCR_EL2	Virtualization Translation Control Register
0b11	0b100	0b0010	0b0010	0b000	VNCR_EL2	Virtualization Control Register
0b11	0b100	0b0010	0b0110	0b000	VSTTBR_EL2	Virtualization Secure Translation Table Base Register
0b11	0b100	0b0010	0b0110	0b010	VSTCR_EL2	Virtualization Secure Translation Control Register
0b11	0b100	0b0011	0b0000	0b000	DACR32_EL2	Domain Control Register
0b11	0b100	0b0011	0b0001	0b100	HDFGRTR_EL2	Hypervisor Debug Filter Grained Trap Register
0b11	0b100	0b0011	0b0001	0b101	HDFGWTR_EL2	Hypervisor Debug Filter Grained Trap Register
0b11	0b100	0b0011	0b0001	0b110	HAFGRTR_EL2	Hypervisor Activity Monitor

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
						Grained Trap Register
0b11	0b100	0b0100	0b0000	0b000	SPSR_EL2	Saved Privileged Status Register (EL2)
0b11	0b100	0b0100	0b0000	0b001	ELR_EL2	Exception Link Register
0b11	0b100	0b0100	0b0001	0b000	SP_EL1	Stack Pointer (EL1)
0b11	0b100	0b0100	0b0011	0b000	SPSR_irq	Saved Privileged Status Register (IRQ mode)
0b11	0b100	0b0100	0b0011	0b001	SPSR_abt	Saved Privileged Status Register (Abort mode)
0b11	0b100	0b0100	0b0011	0b010	SPSR_und	Saved Privileged Status Register (Undefined mode)
0b11	0b100	0b0100	0b0011	0b011	SPSR_fiq	Saved Privileged Status Register (FIQ mode)
0b11	0b100	0b0101	0b0000	0b001	IFSR32_EL2	Instruction Fetch Status Register
0b11	0b100	0b0101	0b0001	0b000	AFSR0_EL2	Auxiliary Status Register 0 (EL2)
0b11	0b100	0b0101	0b0001	0b001	AFSR1_EL2	Auxiliary Status Register 1 (EL2)
0b11	0b100	0b0101	0b0010	0b000	ESR_EL2	Exception Syndrome Register
0b11	0b100	0b0101	0b0010	0b011	VESR_EL2	Virtual Syndrome Register
0b11	0b100	0b0101	0b0011	0b000	FPEXC32_EL2	Floating-Point Exception Control Register
0b11	0b100	0b0101	0b0110	0b000	TFSR_EL2	Tag Fault Status Register (EL2)
0b11	0b100	0b0110	0b0000	0b000	FAR_EL2	Fault Address Register
0b11	0b100	0b0110	0b0000	0b100	HPFAR_EL2	Hypervisor Fault Address Register
0b11	0b100	0b1001	0b1001	0b000	PMSCR_EL2	Statistical Profiling Control Register
0b11	0b100	0b1010	0b0010	0b000	MAIR_EL2	Memory Attribute Indirection Register
0b11	0b100	0b1010	0b0011	0b000	AMAIR_EL2	Auxiliary Memory Attribute Indirection Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b100	0b1010	0b0100	0b000	MPAMHCR_EL2	MPAM Hypervisor Control Register
0b11	0b100	0b1010	0b0100	0b001	MPAMVPMV_EL2	MPAM VPartition Mapping Register
0b11	0b100	0b1010	0b0101	0b000	MPAM2_EL2	MPAM2 Register
0b11	0b100	0b1010	0b0110	0b000	MPAMVPM0_EL2	MPAM VPARTID Mapping Register
0b11	0b100	0b1010	0b0110	0b001	MPAMVPM1_EL2	MPAM VPARTID Mapping Register
0b11	0b100	0b1010	0b0110	0b010	MPAMVPM2_EL2	MPAM VPARTID Mapping Register
0b11	0b100	0b1010	0b0110	0b011	MPAMVPM3_EL2	MPAM VPARTID Mapping Register
0b11	0b100	0b1010	0b0110	0b100	MPAMVPM4_EL2	MPAM VPARTID Mapping Register
0b11	0b100	0b1010	0b0110	0b101	MPAMVPM5_EL2	MPAM VPARTID Mapping Register
0b11	0b100	0b1010	0b0110	0b110	MPAMVPM6_EL2	MPAM VPARTID Mapping Register
0b11	0b100	0b1010	0b0110	0b111	MPAMVPM7_EL2	MPAM VPARTID Mapping Register
0b11	0b100	0b1100	0b0000	0b000	VBAR_EL2	Vector Base Address Register
0b11	0b100	0b1100	0b0000	0b001	RVBAR_EL2	Reset Vector Base Address Register, not implemented
0b11	0b100	0b1100	0b0000	0b010	RMR_EL2	Reset Management Register
0b11	0b100	0b1100	0b0001	0b001	VDISR_EL2	Virtual Deferred Interrupt Status Register
0b11	0b100	0b1100	0b1000	0b0:n[1:0]	ICH_AP0R<n>_EL2	Interrupt Controller Active Priority Group 0 Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b100	0b1100	0b1001	0b0:n[1:0]	ICH_APIR<n>_EL2	Interrupt Controller Active Priority Group 1 Register
0b11	0b100	0b1100	0b1001	0b101	ICC_SRE_EL2	Interrupt Controller System Register register
0b11	0b100	0b1100	0b1011	0b000	ICH_HCR_EL2	Interrupt Controller Control Register
0b11	0b100	0b1100	0b1011	0b001	ICH_VTR_EL2	Interrupt Controller Type Register
0b11	0b100	0b1100	0b1011	0b010	ICH_MISR_EL2	Interrupt Controller Maintenance Interrupt Register
0b11	0b100	0b1100	0b1011	0b011	ICH_EISR_EL2	Interrupt Controller of Interrupt Status Register
0b11	0b100	0b1100	0b1011	0b101	ICH_ELRSR_EL2	Interrupt Controller Empty Local Register Register
0b11	0b100	0b1100	0b1011	0b111	ICH_VMCR_EL2	Interrupt Controller Virtual Machine Control Register
0b11	0b100	0b1100	0b110:n[3]	n[2:0]	ICH_LR<n>_EL2	Interrupt Controller Register
0b11	0b100	0b1101	0b0000	0b001	CONTEXTIDR_EL2	Context Register
0b11	0b100	0b1101	0b0000	0b010	TPIDR_EL2	EL2 Software Thread ID Register
0b11	0b100	0b1101	0b0000	0b111	SCXTNUM_EL2	EL2 Realtime Software Context Number
0b11	0b100	0b1101	0b100:n[3]	n[2:0]	AMEVCNTVOFF0<n>_EL2	Activity Monitor Counter Offset Register 0
0b11	0b100	0b1101	0b101:n[3]	n[2:0]	AMEVCNTVOFF1<n>_EL2	Activity Monitor Counter Offset Register 1
0b11	0b100	0b1110	0b0000	0b011	CNTVOFF_EL2	Counter Virtual Counter register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b100	0b1110	0b0000	0b110	CNTPOFF_EL2	Counter Physical register
0b11	0b100	0b1110	0b0001	0b000	CNTHCTL_EL2	Counter Hypervis Control
0b11	0b100	0b1110	0b0010	0b000	CNTHP_TVAL_EL2	Counter Physical TimerVa register
0b11	0b100	0b1110	0b0010	0b001	CNTHP_CTL_EL2	Counter Hypervis Physical Control
0b11	0b100	0b1110	0b0010	0b010	CNTHP_CVAL_EL2	Counter Physical Compare register
0b11	0b100	0b1110	0b0011	0b000	CNTHV_TVAL_EL2	Counter Virtual T TimerVa Register
0b11	0b100	0b1110	0b0011	0b001	CNTHV_CTL_EL2	Counter Virtual T Control (EL2)
0b11	0b100	0b1110	0b0011	0b010	CNTHV_CVAL_EL2	Counter Virtual T Compare register
0b11	0b100	0b1110	0b0100	0b000	CNTHVS_TVAL_EL2	Counter Secure V Timer TimerVa register
0b11	0b100	0b1110	0b0100	0b001	CNTHVS_CTL_EL2	Counter Secure V Timer Co register
0b11	0b100	0b1110	0b0100	0b010	CNTHVS_CVAL_EL2	Counter Secure V Timer Compare register
0b11	0b100	0b1110	0b0101	0b000	CNTHPS_TVAL_EL2	Counter Secure P Timer TimerVa register
0b11	0b100	0b1110	0b0101	0b001	CNTHPS_CTL_EL2	Counter Secure P Timer Co register
0b11	0b100	0b1110	0b0101	0b010	CNTHPS_CVAL_EL2	Counter Secure P Timer Compare register
0b11	0b110	0b0001	0b0000	0b000	SCTLR_EL3	System Control Register
0b11	0b110	0b0001	0b0000	0b001	ACTLR_EL3	Auxiliary Control Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b11	0b110	0b0001	0b0001	0b000	SCR_EL3	Secure Configuration Register
0b11	0b110	0b0001	0b0001	0b001	SDER32_EL3	AArch32 Secure I/O Enable Register
0b11	0b110	0b0001	0b0001	0b010	CPTR_EL3	Architectural Feature Register
0b11	0b110	0b0001	0b0010	0b000	ZCR_EL3	SVE Control Register
0b11	0b110	0b0001	0b0010	0b110	SMCR_EL3	SME Control Register
0b11	0b110	0b0001	0b0011	0b001	MDCR_EL3	Monitor Configuration Register
0b11	0b110	0b0010	0b0000	0b000	TTBR0_EL3	Translation Table Base Register (EL3)
0b11	0b110	0b0010	0b0000	0b010	TCR_EL3	Translation Control Register
0b11	0b110	0b0010	0b0001	0b100	GPTBR_EL3	Granule Protection Table Base Register
0b11	0b110	0b0010	0b0001	0b110	GPCCR_EL3	Granule Protection Check Control Register
0b11	0b110	0b0100	0b0000	0b000	SPSR_EL3	Saved Program Status Register (EL3)
0b11	0b110	0b0100	0b0000	0b001	ELR_EL3	Exception Link Register
0b11	0b110	0b0100	0b0001	0b000	SP_EL2	Stack Pointer (EL2)
0b11	0b110	0b0101	0b0001	0b000	AFSR0_EL3	Auxiliary Status Register 0 (EL3)
0b11	0b110	0b0101	0b0001	0b001	AFSR1_EL3	Auxiliary Status Register 1 (EL3)
0b11	0b110	0b0101	0b0010	0b000	ESR_EL3	Exception Syndrome Register
0b11	0b110	0b0101	0b0110	0b000	TFSR_EL3	Tag Fault Status Register (EL3)
0b11	0b110	0b0110	0b0000	0b000	FAR_EL3	Fault Address Register
0b11	0b110	0b0110	0b0000	0b101	MFAR_EL3	PA Fault Address Register
0b11	0b110	0b1010	0b0010	0b000	MAIR_EL3	Memory Attribute Indirect Register
0b11	0b110	0b1010	0b0011	0b000	AMAIR_EL3	Auxiliary Memory Attribute Register

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
						Attribute Indirect Register
0b11	0b110	0b1010	0b0101	0b000	MPAM3_EL3	MPAM3 Register
0b11	0b110	0b1100	0b0000	0b000	VBAR_EL3	Vector Base Address Register
0b11	0b110	0b1100	0b0000	0b001	RVBAR_EL3	Reset Vector Base Address Register impleme
0b11	0b110	0b1100	0b0000	0b010	RMR_EL3	Reset Manager Register
0b11	0b110	0b1100	0b1100	0b100	ICC_CTLR_EL3	Interrupt Controller Control Register
0b11	0b110	0b1100	0b1100	0b101	ICC_SRE_EL3	Interrupt Controller System Register register
0b11	0b110	0b1100	0b1100	0b111	ICC_IGRPEN1_EL3	Interrupt Controller Interrupt 1 Enable register
0b11	0b110	0b1101	0b0000	0b010	TPIDR_EL3	EL3 Software Thread ID Register
0b11	0b110	0b1101	0b0000	0b111	SCXTNUM_EL3	EL3 Realtime Software Context Number
0b11	0b111	0b1110	0b0010	0b000	CNTPS_TVAL_EL1	Counter Physical Timer TimerValue register
0b11	0b111	0b1110	0b0010	0b001	CNTPS_CTL_EL1	Counter Physical Timer Control register
0b11	0b111	0b1110	0b0010	0b010	CNTPS_CVAL_EL1	Counter Physical Timer Compare register

Accessed using TLBI:

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
0b01	0b000	0b1000	0b0001	0b000	TLBI VMALLE1OS	TLB Invalidate by VMID, All at stage 1, EL1, Outer Shareable
0b01	0b000	0b1001	0b0001	0b000	TLBI VMALLE1OSNXS	TLB Invalidate by VMID, All at stage 1, EL1, Outer Shareable

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
0b01	0b000	0b1000	0b0001	0b001	TLBI VAE1OS	TLB Invalidate by VA, EL1, Outer Shareable
0b01	0b000	0b1001	0b0001	0b001	TLBI VAE1OSNXS	TLB Invalidate by VA, EL1, Outer Shareable
0b01	0b000	0b1000	0b0001	0b010	TLBI ASIDE1OS	TLB Invalidate by ASID, EL1, Outer Shareable
0b01	0b000	0b1001	0b0001	0b010	TLBI ASIDE1OSNXS	TLB Invalidate by ASID, EL1, Outer Shareable
0b01	0b000	0b1000	0b0001	0b011	TLBI VAAE1OS	TLB Invalidate by VA, All ASID, EL1, Outer Shareable
0b01	0b000	0b1001	0b0001	0b011	TLBI VAAE1OSNXS	TLB Invalidate by VA, All ASID, EL1, Outer Shareable
0b01	0b000	0b1000	0b0001	0b101	TLBI VALE1OS	TLB Invalidate by VA, Last level, EL1, Outer Shareable
0b01	0b000	0b1001	0b0001	0b101	TLBI VALE1OSNXS	TLB Invalidate by VA, Last level, EL1, Outer Shareable
0b01	0b000	0b1000	0b0001	0b111	TLBI VAALE1OS	TLB Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable
0b01	0b000	0b1001	0b0001	0b111	TLBI VAALE1OSNXS	TLB Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable
0b01	0b000	0b1000	0b0010	0b001	TLBI RVAE1IS	TLB Range Invalidate by VA, EL1, Inner Shareable
0b01	0b000	0b1001	0b0010	0b001	TLBI RVAE1ISNXS	TLB Range Invalidate by VA, EL1, Inner Shareable
0b01	0b000	0b1000	0b0010	0b011	TLBI RVAAE1IS	TLB Range Invalidate by VA, All ASID, EL1, Inner Shareable
0b01	0b000	0b1001	0b0010	0b011	TLBI RVAAE1ISNXS	TLB Range Invalidate by VA, All ASID, EL1, Inner Shareable
0b01	0b000	0b1000	0b0010	0b101	TLBI RVALE1IS	TLB Range Invalidate by VA, Last level, EL1, Inner Shareable
0b01	0b000	0b1001	0b0010	0b101	TLBI RVALE1ISNXS	TLB Range Invalidate by VA, Last level, EL1, Inner Shareable
0b01	0b000	0b1000	0b0010	0b111	TLBI RVAALE1IS	TLB Range Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable
0b01	0b000	0b1001	0b0010	0b111	TLBI RVAALE1ISNXS	TLB Range Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable
0b01	0b000	0b1000	0b0011	0b000	TLBI VMALLE1IS	TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable
0b01	0b000	0b1001	0b0011	0b000	TLBI VMALLE1ISNXS	TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
0b01	0b000	0b1000	0b0011	0b001	TLBI VAE1IS	TLB Invalidate by VA, EL1, Inner Shareable
0b01	0b000	0b1001	0b0011	0b001	TLBI VAE1ISNXS	TLB Invalidate by VA, EL1, Inner Shareable
0b01	0b000	0b1000	0b0011	0b010	TLBI ASIDE1IS	TLB Invalidate by ASID, EL1, Inner Shareable
0b01	0b000	0b1001	0b0011	0b010	TLBI ASIDE1ISNXS	TLB Invalidate by ASID, EL1, Inner Shareable
0b01	0b000	0b1000	0b0011	0b011	TLBI VAAE1IS	TLB Invalidate by VA, All ASID, EL1, Inner Shareable
0b01	0b000	0b1001	0b0011	0b011	TLBI VAAE1ISNXS	TLB Invalidate by VA, All ASID, EL1, Inner Shareable
0b01	0b000	0b1000	0b0011	0b101	TLBI VALE1IS	TLB Invalidate by VA, Last level, EL1, Inner Shareable
0b01	0b000	0b1001	0b0011	0b101	TLBI VALE1ISNXS	TLB Invalidate by VA, Last level, EL1, Inner Shareable
0b01	0b000	0b1000	0b0011	0b111	TLBI VAALE1IS	TLB Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable
0b01	0b000	0b1001	0b0011	0b111	TLBI VAALE1ISNXS	TLB Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable
0b01	0b000	0b1000	0b0101	0b001	TLBI RVAE1OS	TLB Range Invalidate by VA, EL1, Outer Shareable
0b01	0b000	0b1001	0b0101	0b001	TLBI RVAE1OSNXS	TLB Range Invalidate by VA, EL1, Outer Shareable
0b01	0b000	0b1000	0b0101	0b011	TLBI RVAAE1OS	TLB Range Invalidate by VA, All ASID, EL1, Outer Shareable
0b01	0b000	0b1001	0b0101	0b011	TLBI RVAAE1OSNXS	TLB Range Invalidate by VA, All ASID, EL1, Outer Shareable
0b01	0b000	0b1000	0b0101	0b101	TLBI RVALE1OS	TLB Range Invalidate by VA, Last level, EL1, Outer Shareable
0b01	0b000	0b1001	0b0101	0b101	TLBI RVALE1OSNXS	TLB Range Invalidate by VA, Last level, EL1, Outer Shareable
0b01	0b000	0b1000	0b0101	0b111	TLBI RVAALE1OS	TLB Range Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable
0b01	0b000	0b1001	0b0101	0b111	TLBI RVAALE1OSNXS	TLB Range Invalidate by VA, All ASID, Last Level, EL1, Outer Shareable
0b01	0b000	0b1000	0b0110	0b001	TLBI RVAE1	TLB Range Invalidate by VA, EL1
0b01	0b000	0b1001	0b0110	0b001	TLBI RVAE1NXS	TLB Range Invalidate by VA, EL1
0b01	0b000	0b1000	0b0110	0b011	TLBI RVAAE1	TLB Range Invalidate by VA, All ASID, EL1

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
0b01	0b000	0b1001	0b0110	0b011	TLBI RVAAE1NXS	TLB Range Invalidate by VA, All ASID, EL1
0b01	0b000	0b1000	0b0110	0b101	TLBI RVALE1	TLB Range Invalidate by VA, Last level, EL1
0b01	0b000	0b1001	0b0110	0b101	TLBI RVALE1NXS	TLB Range Invalidate by VA, Last level, EL1
0b01	0b000	0b1000	0b0110	0b111	TLBI RVAALE1	TLB Range Invalidate by VA, All ASID, Last level, EL1
0b01	0b000	0b1001	0b0110	0b111	TLBI RVAALE1NXS	TLB Range Invalidate by VA, All ASID, Last level, EL1
0b01	0b000	0b1000	0b0111	0b000	TLBI VMALLE1	TLB Invalidate by VMID, All at stage 1, EL1
0b01	0b000	0b1001	0b0111	0b000	TLBI VMALLE1NXS	TLB Invalidate by VMID, All at stage 1, EL1
0b01	0b000	0b1000	0b0111	0b001	TLBI VAE1	TLB Invalidate by VA, EL1
0b01	0b000	0b1001	0b0111	0b001	TLBI VAE1NXS	TLB Invalidate by VA, EL1
0b01	0b000	0b1000	0b0111	0b010	TLBI ASIDE1	TLB Invalidate by ASID, EL1
0b01	0b000	0b1001	0b0111	0b010	TLBI ASIDE1NXS	TLB Invalidate by ASID, EL1
0b01	0b000	0b1000	0b0111	0b011	TLBI VAAE1	TLB Invalidate by VA, All ASID, EL1
0b01	0b000	0b1001	0b0111	0b011	TLBI VAAE1NXS	TLB Invalidate by VA, All ASID, EL1
0b01	0b000	0b1000	0b0111	0b101	TLBI VALE1	TLB Invalidate by VA, Last level, EL1
0b01	0b000	0b1001	0b0111	0b101	TLBI VALE1NXS	TLB Invalidate by VA, Last level, EL1
0b01	0b000	0b1000	0b0111	0b111	TLBI VAALE1	TLB Invalidate by VA, All ASID, Last level, EL1
0b01	0b000	0b1001	0b0111	0b111	TLBI VAALE1NXS	TLB Invalidate by VA, All ASID, Last level, EL1
0b01	0b100	0b1000	0b0000	0b001	TLBI IPAS2E1IS	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
0b01	0b100	0b1001	0b0000	0b001	TLBI IPAS2E1ISNXS	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
0b01	0b100	0b1000	0b0000	0b010	TLBI RIPAS2E1IS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
0b01	0b100	0b1001	0b0000	0b010	TLBI RIPAS2E1ISNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
0b01	0b100	0b1000	0b0000	0b101	TLBI IPAS2LE1IS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
						level, EL1, Inner Shareable
0b01	0b100	0b1001	0b0000	0b101	TLBI IPAS2LE1ISNXS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
0b01	0b100	0b1000	0b0000	0b110	TLBI RIPAS2LE1IS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
0b01	0b100	0b1001	0b0000	0b110	TLBI RIPAS2LE1ISNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
0b01	0b100	0b1000	0b0001	0b000	TLBI ALLE2OS	TLB Invalidate All, EL2, Outer Shareable
0b01	0b100	0b1001	0b0001	0b000	TLBI ALLE2OSNXS	TLB Invalidate All, EL2, Outer Shareable
0b01	0b100	0b1000	0b0001	0b001	TLBI VAE2OS	TLB Invalidate by VA, EL2, Outer Shareable
0b01	0b100	0b1001	0b0001	0b001	TLBI VAE2OSNXS	TLB Invalidate by VA, EL2, Outer Shareable
0b01	0b100	0b1000	0b0001	0b100	TLBI ALLE1OS	TLB Invalidate All, EL1, Outer Shareable
0b01	0b100	0b1001	0b0001	0b100	TLBI ALLE1OSNXS	TLB Invalidate All, EL1, Outer Shareable
0b01	0b100	0b1000	0b0001	0b101	TLBI VALE2OS	TLB Invalidate by VA, Last level, EL2, Outer Shareable
0b01	0b100	0b1001	0b0001	0b101	TLBI VALE2OSNXS	TLB Invalidate by VA, Last level, EL2, Outer Shareable
0b01	0b100	0b1000	0b0001	0b110	TLBI VMALLS12E1OS	TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Outer Shareable
0b01	0b100	0b1001	0b0001	0b110	TLBI VMALLS12E1OSNXS	TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Outer Shareable
0b01	0b100	0b1000	0b0010	0b001	TLBI RVAE2IS	TLB Range Invalidate by VA, EL2, Inner Shareable
0b01	0b100	0b1001	0b0010	0b001	TLBI RVAE2ISNXS	TLB Range Invalidate by VA, EL2, Inner Shareable
0b01	0b100	0b1000	0b0010	0b101	TLBI RVALE2IS	TLB Range Invalidate by VA, Last level, EL2, Inner Shareable
0b01	0b100	0b1001	0b0010	0b101	TLBI RVALE2ISNXS	TLB Range Invalidate by VA, Last level, EL2, Inner Shareable
0b01	0b100	0b1000	0b0011	0b000	TLBI ALLE2IS	TLB Invalidate All, EL2, Inner Shareable
0b01	0b100	0b1001	0b0011	0b000	TLBI ALLE2ISNXS	TLB Invalidate All, EL2, Inner Shareable
0b01	0b100	0b1000	0b0011	0b001	TLBI VAE2IS	TLB Invalidate by VA, EL2, Inner Shareable

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
0b01	0b100	0b1001	0b0011	0b001	TLBI VAE2ISNXS	TLB Invalidate by VA, EL2, Inner Shareable
0b01	0b100	0b1000	0b0011	0b100	TLBI ALLE1IS	TLB Invalidate All, EL1, Inner Shareable
0b01	0b100	0b1001	0b0011	0b100	TLBI ALLE1ISNXS	TLB Invalidate All, EL1, Inner Shareable
0b01	0b100	0b1000	0b0011	0b101	TLBI VALE2IS	TLB Invalidate by VA, Last level, EL2, Inner Shareable
0b01	0b100	0b1001	0b0011	0b101	TLBI VALE2ISNXS	TLB Invalidate by VA, Last level, EL2, Inner Shareable
0b01	0b100	0b1000	0b0011	0b110	TLBI VMALLS12E1IS	TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner Shareable
0b01	0b100	0b1001	0b0011	0b110	TLBI VMALLS12E1ISNXS	TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner Shareable
0b01	0b100	0b1000	0b0100	0b000	TLBI IPAS2E1OS	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
0b01	0b100	0b1001	0b0100	0b000	TLBI IPAS2E1OSNXS	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
0b01	0b100	0b1000	0b0100	0b001	TLBI IPAS2E1	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1
0b01	0b100	0b1001	0b0100	0b001	TLBI IPAS2E1NXS	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1
0b01	0b100	0b1000	0b0100	0b010	TLBI RIPAS2E1	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1
0b01	0b100	0b1001	0b0100	0b010	TLBI RIPAS2E1NXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1
0b01	0b100	0b1000	0b0100	0b011	TLBI RIPAS2E1OS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
0b01	0b100	0b1001	0b0100	0b011	TLBI RIPAS2E1OSNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, EL1, Outer Shareable
0b01	0b100	0b1000	0b0100	0b100	TLBI IPAS2LE1OS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
0b01	0b100	0b1001	0b0100	0b100	TLBI IPAS2LE1OSNXS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
0b01	0b100	0b1000	0b0100	0b101	TLBI IPAS2LE1	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
0b01	0b100	0b1001	0b0100	0b101	TLBI IPAS2LE1NXS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
0b01	0b100	0b1000	0b0100	0b110	TLBI RIPAS2LE1	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
0b01	0b100	0b1001	0b0100	0b110	TLBI RIPAS2LE1NXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
0b01	0b100	0b1000	0b0100	0b111	TLBI RIPAS2LE1OS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
0b01	0b100	0b1001	0b0100	0b111	TLBI RIPAS2LE1OSNXS	TLB Range Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Outer Shareable
0b01	0b100	0b1000	0b0101	0b001	TLBI RVAE2OS	TLB Range Invalidate by VA, EL2, Outer Shareable
0b01	0b100	0b1001	0b0101	0b001	TLBI RVAE2OSNXS	TLB Range Invalidate by VA, EL2, Outer Shareable
0b01	0b100	0b1000	0b0101	0b101	TLBI RVALE2OS	TLB Range Invalidate by VA, Last level, EL2, Outer Shareable
0b01	0b100	0b1001	0b0101	0b101	TLBI RVALE2OSNXS	TLB Range Invalidate by VA, Last level, EL2, Outer Shareable
0b01	0b100	0b1000	0b0110	0b001	TLBI RVAE2	TLB Range Invalidate by VA, EL2
0b01	0b100	0b1001	0b0110	0b001	TLBI RVAE2NXS	TLB Range Invalidate by VA, EL2
0b01	0b100	0b1000	0b0110	0b101	TLBI RVALE2	TLB Range Invalidate by VA, Last level, EL2
0b01	0b100	0b1001	0b0110	0b101	TLBI RVALE2NXS	TLB Range Invalidate by VA, Last level, EL2
0b01	0b100	0b1000	0b0111	0b000	TLBI ALLE2	TLB Invalidate All, EL2
0b01	0b100	0b1001	0b0111	0b000	TLBI ALLE2NXS	TLB Invalidate All, EL2
0b01	0b100	0b1000	0b0111	0b001	TLBI VAE2	TLB Invalidate by VA, EL2
0b01	0b100	0b1001	0b0111	0b001	TLBI VAE2NXS	TLB Invalidate by VA, EL2
0b01	0b100	0b1000	0b0111	0b100	TLBI ALLE1	TLB Invalidate All, EL1
0b01	0b100	0b1001	0b0111	0b100	TLBI ALLE1NXS	TLB Invalidate All, EL1
0b01	0b100	0b1000	0b0111	0b101	TLBI VALE2	TLB Invalidate by VA, Last level, EL2
0b01	0b100	0b1001	0b0111	0b101	TLBI VALE2NXS	TLB Invalidate by VA, Last level, EL2

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
0b01	0b100	0b1000	0b0111	0b110	TLBI VMALLS12E1	TLB Invalidate by VMID, All at Stage 1 and 2, EL1
0b01	0b100	0b1001	0b0111	0b110	TLBI VMALLS12E1NXS	TLB Invalidate by VMID, All at Stage 1 and 2, EL1
0b01	0b110	0b1000	0b0001	0b000	TLBI ALLE3OS	TLB Invalidate All, EL3, Outer Shareable
0b01	0b110	0b1001	0b0001	0b000	TLBI ALLE3OSNXS	TLB Invalidate All, EL3, Outer Shareable
0b01	0b110	0b1000	0b0001	0b001	TLBI VAE3OS	TLB Invalidate by VA, EL3, Outer Shareable
0b01	0b110	0b1001	0b0001	0b001	TLBI VAE3OSNXS	TLB Invalidate by VA, EL3, Outer Shareable
0b01	0b110	0b1000	0b0001	0b100	TLBI PAALLOS	TLB Invalidate GPT Information by PA, All Entries, Outer Shareable
0b01	0b110	0b1000	0b0001	0b101	TLBI VALE3OS	TLB Invalidate by VA, Last level, EL3, Outer Shareable
0b01	0b110	0b1001	0b0001	0b101	TLBI VALE3OSNXS	TLB Invalidate by VA, Last level, EL3, Outer Shareable
0b01	0b110	0b1000	0b0010	0b001	TLBI RVAE3IS	TLB Range Invalidate by VA, EL3, Inner Shareable
0b01	0b110	0b1001	0b0010	0b001	TLBI RVAE3ISNXS	TLB Range Invalidate by VA, EL3, Inner Shareable
0b01	0b110	0b1000	0b0010	0b101	TLBI RVALE3IS	TLB Range Invalidate by VA, Last level, EL3, Inner Shareable
0b01	0b110	0b1001	0b0010	0b101	TLBI RVALE3ISNXS	TLB Range Invalidate by VA, Last level, EL3, Inner Shareable
0b01	0b110	0b1000	0b0011	0b000	TLBI ALLE3IS	TLB Invalidate All, EL3, Inner Shareable
0b01	0b110	0b1001	0b0011	0b000	TLBI ALLE3ISNXS	TLB Invalidate All, EL3, Inner Shareable
0b01	0b110	0b1000	0b0011	0b001	TLBI VAE3IS	TLB Invalidate by VA, EL3, Inner Shareable
0b01	0b110	0b1001	0b0011	0b001	TLBI VAE3ISNXS	TLB Invalidate by VA, EL3, Inner Shareable
0b01	0b110	0b1000	0b0011	0b101	TLBI VALE3IS	TLB Invalidate by VA, Last level, EL3, Inner Shareable
0b01	0b110	0b1001	0b0011	0b101	TLBI VALE3ISNXS	TLB Invalidate by VA, Last level, EL3, Inner Shareable
0b01	0b110	0b1000	0b0100	0b011	TLBI RPAOS	TLB Range Invalidate GPT Information by PA, Outer Shareable
0b01	0b110	0b1000	0b0100	0b111	TLBI RPALOS	TLB Range Invalidate GPT Information by PA, Last level, Outer Shareable
0b01	0b110	0b1000	0b0101	0b001	TLBI RVAE3OS	TLB Range Invalidate by VA, EL3, Outer Shareable

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
0b01	0b110	0b1001	0b0101	0b001	TLBI RVAE3OSNXS	TLB Range Invalidate by VA, EL3, Outer Shareable
0b01	0b110	0b1000	0b0101	0b101	TLBI RVALE3OS	TLB Range Invalidate by VA, Last level, EL3, Outer Shareable
0b01	0b110	0b1001	0b0101	0b101	TLBI RVALE3OSNXS	TLB Range Invalidate by VA, Last level, EL3, Outer Shareable
0b01	0b110	0b1000	0b0110	0b001	TLBI RVAE3	TLB Range Invalidate by VA, EL3
0b01	0b110	0b1001	0b0110	0b001	TLBI RVAE3NXS	TLB Range Invalidate by VA, EL3
0b01	0b110	0b1000	0b0110	0b101	TLBI RVALE3	TLB Range Invalidate by VA, Last level, EL3
0b01	0b110	0b1001	0b0110	0b101	TLBI RVALE3NXS	TLB Range Invalidate by VA, Last level, EL3
0b01	0b110	0b1000	0b0111	0b000	TLBI ALLE3	TLB Invalidate All, EL3
0b01	0b110	0b1001	0b0111	0b000	TLBI ALLE3NXS	TLB Invalidate All, EL3
0b01	0b110	0b1000	0b0111	0b001	TLBI VAE3	TLB Invalidate by VA, EL3
0b01	0b110	0b1001	0b0111	0b001	TLBI VAE3NXS	TLB Invalidate by VA, EL3
0b01	0b110	0b1000	0b0111	0b100	TLBI PAALL	TLB Invalidate GPT Information by PA, All Entries, Local
0b01	0b110	0b1000	0b0111	0b101	TLBI VALE3	TLB Invalidate by VA, Last level, EL3
0b01	0b110	0b1001	0b0111	0b101	TLBI VALE3NXS	TLB Invalidate by VA, Last level, EL3

3020/09/2021 1412:5740

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

System 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_AA64PFR0_EL1	AArch64 Processor Feature Register 0
AArch64	ID_AA64PFR1_EL1	AArch64 Processor Feature Register 1
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

Exec state	Name	Description
AArch32	AMAIR1	Auxiliary Memory Attribute Indirection Register 1
AArch32	CONTEXTIDR	Context ID Register
AArch32	DACR	Domain Access Control Register
AArch32	HAMAIR0	Hyp Auxiliary Memory Attribute Indirection Register 0
AArch32	HAMAIR1	Hyp Auxiliary Memory Attribute Indirection Register 1
AArch32	HMAIR0	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	MAIR0	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	VTCR	Virtualization Translation Control Register
AArch32	VTTBR	Virtualization Translation Table Base 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	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	MAIR_EL1	Memory Attribute Indirection Register (EL1)
AArch64	MAIR_EL2	Memory Attribute Indirection Register (EL2)
AArch64	MAIR_EL3	Memory Attribute Indirection Register (EL3)
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)
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	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	PA Fault Address Register
AArch64	VBAR_EL1	Vector Base Address Register (EL1)
AArch64	VBAR_EL2	Vector Base Address Register (EL2)
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)

Exec state	Name	Description
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	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
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 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 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

Exec state	Name	Description
AArch64	DC IGSW	Invalidate of Allocation Tags by Set/Way
AArch64	DC IGVC	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
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	CPRCTX	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

Exec state	Name	Description
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	TLBIMVAAAL	TLB Invalidate by VA, All ASID, Last level
AArch32	TLBIMVAAALIS	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
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

Exec state	Name	Description
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
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

Exec state	Name	Description
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

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	PMEVTYPEPER<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
AArch32	PMXEVCNTR	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	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	PMEVTYPEPER<n>_EL0	Performance Monitors Event Type Registers
AArch64	PMINTENCLR_EL1	Performance Monitors Interrupt Enable Clear register
AArch64	PMINTENSET_EL1	Performance Monitors Interrupt Enable Set register

Exec state	Name	Description
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	PMCCNTR_EL0	Performance Monitors Cycle Counter
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	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	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	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	PMEVFILTR<n>	Performance Monitors Event Type Select Register <n>
External	PMEVTYPER<n>_EL0	Performance Monitors Event Type Registers
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
External	PMLSR	Performance Monitors Lock Status Register
External	PMMIR	Performance Monitors Machine Identification Register
External	PMOVSCCLR_EL0	Performance Monitors Overflow Flag Status Clear register
External	PMOVSSSET_EL0	Performance Monitors Overflow Flag Status Set 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	PMSWINC_EL0	Performance Monitors Software Increment 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
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>	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
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	CNTELOACR	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

Exec state	Name	Description
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
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)

Exec state	Name	Description
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
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	EDECCR	External Debug Exception Catch Control Register
External	EDECR	External Debug Execution Control Register
External	EDES	External Debug Event Status 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	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	CTIAPPPULSE	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

Exec state	Name	Description
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
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

Exec state	Name	Description
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
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	CNTHCTL_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	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	SCTLR_EL2	System 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_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

Exec state	Name	Description
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 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	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	FPSR	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	TRCAUXCLR	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
AArch64	TRCIMSPEC<n>	IMP DEF Register <n>
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	TRCSTALLCTLR	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

Exec state	Name	Description
AArch64	TRCVIPCSSLTLR	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	TRCAUXCTLR	Auxiliary Control Register
External	TRCBBCTLR	Branch Broadcast Control Register
External	TRCCCCTLR	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	TRCCNTCTLR<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	TRCEXTINSELR<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	TRCPRGCTLR	Programming Control Register
External	TRCQCTLR	Q Element Control Register
External	TRCRSCTLR<n>	Resource Selection Control Register <n>
External	TRCRSR	Resources Status Register
External	TRCSEQEVR<n>	Sequencer State Transition Control Register <n>
External	TRCSEQRSTEV	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	TRSTALLCTLR	Stall Control Register
External	TRCSTATR	Trace Status Register
External	TRCSYNCP	Synchronization Period Register
External	TRCTRACEIDR	Trace ID Register
External	TRCTSCTLR	Timestamp Control Register
External	TRCVICTLR	ViewInst Main Control Register
External	TRCVIIECTLR	ViewInst Include/Exclude Control Register
External	TRCVIPCSSLTLR	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
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

Exec state	Name	Description
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_NMIAR1_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_NMIAR1_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>	SPI 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

Exec state	Name	Description
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_ISENBALER<n>	Interrupt Set-Enable Registers
External	GICD_ISENBALER<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
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

Exec state	Name	Description
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
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
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 Address Register
External	ERR<n>CTLR	Error Record Control Register
External	ERR<n>FR	Error Record Feature Register
External	ERR<n>MISC0	Error Record Miscellaneous Register 0
External	ERR<n>MISC1	Error Record Miscellaneous Register 1
External	ERR<n>MISC2	Error Record Miscellaneous Register 2
External	ERR<n>MISC3	Error Record Miscellaneous Register 3

Exec state	Name	Description
External	ERR<n>PFGCDN	Pseudo-fault Generation Countdown Register
External	ERR<n>PFGCTL	Pseudo-fault Generation Control Register
External	ERR<n>PFGF	Pseudo-fault Generation Feature Register
External	ERR<n>STATUS	Error Record 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	ERRIMPDEF<n>	IMPLEMENTATION DEFINED Register <n>
External	ERRIRQCR<n>	Generic Error Interrupt Configuration Register
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)
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_CPBM<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

Exec state	Name	Description
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_EVNT	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
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])

Exec state	Name	Description
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
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	GPCCR_EL3	Granule Protection Check Control Register (EL3)
AArch64	GPTBR_EL3	Granule Protection Table Base Register
AArch64	MFAR_EL3	PA Fault Address Register

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

3020/09/2021 1412:5740

Copyright © 2010-2021 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

[CNTELOACR](#): 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

EDECCR: External Debug Exception Catch Control Register

EDECR: External Debug Execution Control Register

[EDES](#): External Debug Event 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

[EDVIDSR](#): External Debug Virtual Context Sample Register

EDWAR: External Debug Watchpoint Address Register

ERR<n>ADDR: Error Record Address Register

ERR<n>CTLR: Error Record Control Register

ERR<n>FR: Error Record Feature Register

ERR<n>MISC0: Error Record Miscellaneous Register 0

ERR<n>MISC1: Error Record Miscellaneous Register 1

ERR<n>MISC2: Error Record Miscellaneous Register 2

ERR<n>MISC3: Error Record Miscellaneous Register 3

ERR<n>PFGCDN: Pseudo-fault Generation Countdown Register

ERR<n>PFGCTL: Pseudo-fault Generation Control Register

ERR<n>PFGF: Pseudo-fault Generation Feature Register

ERR<n>STATUS: Error Record 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

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_IGRPMODR<n>](#): Interrupt Group Modifier Registers

[GICD_IGRPMODR<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_CPB<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

PMCCNTR_EL0: Performance Monitors Cycle Counter

[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

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

PMCNTENCLR_EL0: Performance Monitors Count Enable Clear register

PMCNTENSET_EL0: Performance Monitors Count Enable Set register

[PMCR_EL0](#): Performance Monitors Control 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

[PMEVFILTR<n>](#): Performance Monitors Event Type Select Register <n>

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

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

PMOVSLR_EL0: Performance Monitors Overflow Flag Status Clear register

PMOVSET_EL0: Performance Monitors Overflow Flag Status Set 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

PMSWINC_EL0: Performance Monitors Software Increment register

PMVIDSR: VMID Sample 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

TRCDEVAFF: 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

TRCEXTINSEL<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

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>

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>

TRCSTALLCTL: Stall Control Register

TRCSTATR: Trace Status Register

TRCSYNCP: Synchronization Period Register

TRCTRACEIDR: Trace ID Register

TRCTSCTL: Timestamp Control Register

TRCVICTL: ViewInst Main Control Register

TRCVIIECTL: ViewInst Include/Exclude Control Register

TRCVIPCSSL: ViewInst Start/Stop PE Comparator Control Register

TRCVISSCTL: ViewInst Start/Stop Control Register

TRCVMIDCCTL0: Virtual Context Identifier Comparator Control Register 0

TRCVMIDCCTL1: Virtual Context Identifier Comparator Control Register 1

TRCVMIDCVR<n>: Virtual Context Identifier Comparator Value Register <n>

3020/09/2021 1412:5740

Copyright © 2010-2021 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:

- [GIC Distributor](#)
- [Debug](#)
- [GIC Virtual interface control](#)
- [PMU](#)
- [GIC Redistributor](#) [GIC Virtual CPU interface](#)
- [GIC Virtual CPU interface](#) [GIC Redistributor](#)
- [CTI](#)
- [GIC ITS control](#)
- [GIC CPU interface](#)
- [Timer](#)
- [GIC ITS translation](#)
- [AMU](#)
- [ETE](#)
- [MPAM](#)
- [RAS](#)

In the GIC Distributor block:

Frame	Offset	Name	Description
Dist_base	0x0000	GICD_CTLR	Distributor Control Register
Dist_base	0x0004	GICD_TYPER	Interrupt Controller Type Register
Dist_base	0x0008	GICD_IIDR	Distributor Implementer Identification Register
Dist_base	0x000C	GICD_TYPER2	Interrupt Controller Type Register 2
Dist_base	0x0010	GICD_STATUSR	Error Reporting Status Register
Dist_base	0x0010	GICD_STATUSR	Error Reporting Status Register
Dist_base	0x0040	GICD_SETSPI_NSR	Set Non-secure SPI Pending Register
Dist_base	0x0048	GICD_CLRSPI_NSR	Clear Non-secure SPI Pending Register
Dist_base	0x0050	GICD_SETSPI_SR	Set Secure SPI Pending Register
Dist_base	0x0058	GICD_CLRSPI_SR	Clear Secure SPI Pending Register
Dist_base	0x0080 + (4 * n)	GICD_IGROUPR<n>	Interrupt Group Registers
Dist_base	0x0100 + (4 * n)	GICD_ISENBALER<n>	Interrupt Set-Enable Registers
Dist_base	0x0180 + (4 * n)	GICD_ICENABLER<n>	Interrupt Clear-Enable Registers
Dist_base	0x0200 + (4 * n)	GICD_ISPENDR<n>	Interrupt Set-Pending Registers
Dist_base	0x0280 + (4 * n)	GICD_ICPENDR<n>	Interrupt Clear-Pending Registers
Dist_base	0x0300 + (4 * n)	GICD_ISACTIVER<n>	Interrupt Set-Active Registers
Dist_base	0x0380 + (4 * n)	GICD_ICACTIVER<n>	Interrupt Clear-Active Registers
Dist_base	0x0400 + (4 * n)	GICD_IPRIORITYR<n>	Interrupt Priority Registers
Dist_base	0x0800 + (4 * n)	GICD_ITARGETSR<n>	Interrupt Processor Targets Registers
Dist_base	0x0C00 + (4 * n)	GICD_ICFGR<n>	Interrupt Configuration Registers
Dist_base	0x0D00 + (4 * n)	GICD_IGRPMODR<n>	Interrupt Group Modifier Registers
Dist_base	0x0E00 + (4 * n)	GICD_NSACR<n>	Non-secure Access Control Registers
Dist_base	0x0F00	GICD_SGIR	Software Generated Interrupt Register
Dist_base	0x0F10 + (4 * n)	GICD_CPENDSGIR<n>	Sgi Clear-Pending Registers
Dist_base	0x0F20 + (4 * n)	GICD_SPENDSGIR<n>	Sgi Set-Pending Registers
Dist_base	0x0F80 + (4 * n)	GICD_INMIR<n>	Non-maskable Interrupt Registers, x = 0 to 31
Dist_base	0x1000 + (4 * n)	GICD_IGROUPR<n>E	Interrupt Group Registers (extended SPI range)

Frame	Offset	Name	Description
Dist_base	0x1200 + (4 * n)	GICD_ISENBALER<n>E	Interrupt Set-Enable Registers
Dist_base	0x1400 + (4 * n)	GICD_ICENABLER<n>E	Interrupt Clear-Enable Registers
Dist_base	0x1600 + (4 * n)	GICD_ISPENDR<n>E	Interrupt Set-Pending Registers (extended SPI range)
Dist_base	0x1800 + (4 * n)	GICD_ICPENDR<n>E	Interrupt Clear-Pending Registers (extended SPI range)
Dist_base	0x1A00 + (4 * n)	GICD_ISACTIVER<n>E	Interrupt Set-Active Registers (extended SPI range)
Dist_base	0x1C00 + (4 * n)	GICD_ICACTIVER<n>E	Interrupt Clear-Active Registers (extended SPI range)
Dist_base	0x2000 + (4 * n)	GICD_IPRIORITYR<n>E	Holds the priority of the corresponding interrupt for each extended SPI supported by the GIC.
Dist_base	0x3000 + (4 * n)	GICD_ICFGR<n>E	Interrupt Configuration Registers (Extended SPI Range)
Dist_base	0x3400 + (4 * n)	GICD_IGRPMODR<n>E	Interrupt Group Modifier Registers (extended SPI range)
Dist_base	0x3600 + (4 * n)	GICD_NSACR<n>E	Non-secure Access Control Registers
Dist_base	0x3B00 + (4 * n)	GICD_INMIR<n>E	Non-maskable Interrupt Registers for Extended SPIs, x = 0 to 31
Dist_base	0x6000 + (8 * n)	GICD_IROUTER<n>	Interrupt Routing Registers
Dist_base	0x8000 + (8 * n)	GICD_IROUTER<n>E	Interrupt Routing Registers (Extended SPI Range)
MSI_base	0x0004	GICM_TYPER	Distributor MSI Type Register
MSI_base	0x0040	GICM_SETSPI_NSR	Set Non-secure SPI Pending Register
MSI_base	0x0048	GICM_CLRSPI_NSR	Clear Non-secure SPI Pending Register
MSI_base	0x0050	GICM_SETSPI_SR	Set Secure SPI Pending Register
MSI_base	0x0058	GICM_CLRSPI_SR	Clear Secure SPI Pending Register
MSI_base	0x0FCC	GICM_IIDR	Distributor Implementer Identification Register

In the Debug block:

Offset	Name	Description
0x020	EDES	External Debug Event Status Register
0x024	EDEC	External Debug Execution Control Register
0x030	EDWAR[31:0]	External Debug Watchpoint Address Register
0x034	EDWAR[63:32]	External Debug Watchpoint Address Register
0x080	DBGDTRRX_EL0	Debug Data Transfer Register, Receive
0x084	EDITR	External Debug Instruction Transfer Register
0x088	EDSCR	External Debug Status and Control Register
0x08C	DBGDTRTX_EL0	Debug Data Transfer Register, Transmit
0x090	EDRCR	External Debug Reserve Control Register
0x094	EDACR	External Debug Auxiliary Control Register
0x098	EDECCR	External Debug Exception Catch Control Register
0x0A0	EDPCSR[31:0]	External Debug Program Counter Sample Register
0x0A4	EDCIDS	External Debug Context ID Sample Register
0x0A8	EDVIDSR	External Debug Virtual Context Sample Register
0x0AC	EDPCSR[63:32]	External Debug Program Counter Sample Register
0x300	OSLAR_EL1	OS Lock Access Register
0x310	EDPRCR	External Debug Power/Reset Control Register
0x314	EDPRSR	External Debug Processor Status Register
0x400 + (16 * n)	DBGBVR<n>_EL1[63:0]	Debug Breakpoint Value Registers

Offset	Name	Description
0x408 + (16 * n)	DBGBCR<n>_EL1	Debug Breakpoint Control Registers
0x800 + (16 * n)	DBGWVR<n>_EL1[63:0]	Debug Watchpoint Value Registers
0x808 + (16 * n)	DBGWCR<n>_EL1	Debug Watchpoint Control Registers
0xD00	MIDR_EL1	Main ID Register
0xD20	EDPFR[31:0]	External Debug Processor Feature Register
0xD24	EDPFR[63:32]	External Debug Processor Feature Register
0xD28	EDDFR[31:0]	External Debug Feature Register
0xD2C	EDDFR[63:32]	External Debug Feature Register
0xD60	EDAA32PFR	External Debug Auxiliary Processor Feature Register
0xF00	EDITCTRL	External Debug Integration mode Control register
0xFA0	DBGCLAIMSET_EL1	Debug CLAIM Tag Set register
0xFA4	DBGCLAIMCLR_EL1	Debug CLAIM Tag Clear register
0xFA8	EDDEVAFF0	External Debug Device Affinity register 0
0xFAC	EDDEVAFF1	External Debug Device Affinity register 1
0xFB0	EDLAR	External Debug Lock Access Register
0xFB4	EDLSR	External Debug Lock Status Register
0xFB8	DBGAUTHSTATUS_EL1	Debug Authentication Status register
0xFBC	EDDEVARCH	External Debug Device Architecture register
0xFC0	EDDEVID2	External Debug Device ID register 2
0xFC4	EDDEVID1	External Debug Device ID register 1
0xFC8	EDDEVID	External Debug Device ID register 0
0xFCC	EDDEVTYPE	External Debug Device Type register
0xFD0	EDPIDR4	External Debug Peripheral Identification Register 4
0xFE0	EDPIDR0	External Debug Peripheral Identification Register 0
0xFE4	EDPIDR1	External Debug Peripheral Identification Register 1
0xFE8	EDPIDR2	External Debug Peripheral Identification Register 2
0xFEC	EDPIDR3	External Debug Peripheral Identification Register 3
0xFF0	EDCIDR0	External Debug Component Identification Register 0
0xFF4	EDCIDR1	External Debug Component Identification Register 1
0xFF8	EDCIDR2	External Debug Component Identification Register 2
0xFFC	EDCIDR3	External Debug Component Identification Register 3

In the GIC Virtual interface control block:

Offset	Name	Description
0x0000	GICH_HCR	Hypervisor Control Register
0x0004	GICH_VTR	Virtual Type Register
0x0008	GICH_VMCR	Virtual Machine Control Register
0x0010	GICH_MISR	Maintenance Interrupt Status Register
0x0020	GICH_EISR	End Interrupt Status Register
0x0030	GICH_ELRSR	Empty List Register Status Register
0x00F0 + (4 * n)	GICH_APR<n>	Active Priorities Registers
0x0100 + (4 * n)	GICH_LR<n>	List Registers

In the PMU block:

Offset	Name	Description
0x000 + (8 * n)	PMEVCNTR<n>_EL0	Performance Monitors Event Count Registers
0x0F8	PMCCNTR_EL0[31:0]	Performance Monitors Cycle Counter

Offset	Name	Description
0x0FC	PMCCNTR_EL0[63:32]	Performance Monitors Cycle Counter
0x200	PMPCSR[31:0]	Program Counter Sample Register
0x204	PMPCSR[63:32]	Program Counter Sample Register
0x208	PMCID1SR	CONTEXTIDR_EL1 Sample Register
0x20C	PMVIDSR	VMID Sample Register
0x220	PMPCSR[31:0]	Program Counter Sample Register
0x224	PMPCSR[63:32]	Program Counter Sample Register
0x228	PMCID1SR	CONTEXTIDR_EL1 Sample Register
0x22C	PMCID2SR	CONTEXTIDR_EL2 Sample Register
0x400 + (4 * n)	PMEVTYPEPER<n>_EL0	Performance Monitors Event Type Registers
0x47C	PMCCFILTR_EL0	Performance Monitors Cycle Counter Filter Register
0xA00 + (4 * n)	PMEVFILTR<n>	Performance Monitors Event Type Select Register &lt;n>
0xC00	PMCNTENSET_EL0	Performance Monitors Count Enable Set register
0xC20	PMCNTENCLR_EL0	Performance Monitors Count Enable Clear register
0xC40	PMINTENSET_EL1	Performance Monitors Interrupt Enable Set register
0xC60	PMINTENCLR_EL1	Performance Monitors Interrupt Enable Clear register
0xC80	PMOVSLR_EL0	Performance Monitors Overflow Flag Status Clear register
0xCA0	PMSWINC_EL0	Performance Monitors Software Increment register
0xCC0	PMOVSSET_EL0	Performance Monitors Overflow Flag Status Set register
0xE00	PMCFGR	Performance Monitors Configuration Register
0xE04	PMCR_EL0	Performance Monitors Control Register
0xE20	PMCEID0	Performance Monitors Common Event Identification register 0
0xE24	PMCEID1	Performance Monitors Common Event Identification register 1
0xE28	PMCEID2	Performance Monitors Common Event Identification register 2
0xE2C	PMCEID3	Performance Monitors Common Event Identification register 3
0xE40	PMMIR	Performance Monitors Machine Identification Register
0xF00	PMITCTRL	Performance Monitors Integration mode Control register
0xFA8	PMDEVAFF0	Performance Monitors Device Affinity register 0
0xFAC	PMDEVAFF1	Performance Monitors Device Affinity register 1
0xFB0	PMLAR	Performance Monitors Lock Access Register
0xFB4	PMLSR	Performance Monitors Lock Status Register
0xFB8	PMAUTHSTATUS	Performance Monitors Authentication Status register
0xFBC	PMDEVARCH	Performance Monitors Device Architecture register
0xFC8	PMDEVID	Performance Monitors Device ID register
0xFCC	PMDEVTYPE	Performance Monitors Device Type register
0xFD0	PMPIDR4	Performance Monitors Peripheral Identification Register 4
0xFE0	PMPIDR0	Performance Monitors Peripheral Identification Register 0
0xFE4	PMPIDR1	Performance Monitors Peripheral Identification Register 1
0xFE8	PMPIDR2	Performance Monitors Peripheral Identification Register 2
0xFEC	PMPIDR3	Performance Monitors Peripheral Identification Register 3
0xFF0	PMCIDR0	Performance Monitors Component Identification Register 0
0xFF4	PMCIDR1	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	Performance Monitors Component Identification Register 3

In the GIC Virtual CPU interface block:

Offset	Name	Description
0x0000	GICV_CTLR	Virtual Machine Control Register
0x0004	GICV_PMR	Virtual Machine Priority Mask Register
0x0008	GICV_BPR	Virtual Machine Binary Point Register
0x000C	GICV_IAR	Virtual Machine Interrupt Acknowledge Register
0x0010	GICV_EOIR	Virtual Machine End Of Interrupt Register
0x0014	GICV_RPR	Virtual Machine Running Priority Register
0x0018	GICV_HPPIR	Virtual Machine Highest Priority Pending Interrupt Register
0x001C	GICV_ABPR	Virtual Machine Aliased Binary Point Register
0x0020	GICV_AIAR	Virtual Machine Aliased Interrupt Acknowledge Register
0x0024	GICV_AEOIR	Virtual Machine Aliased End Of Interrupt Register
0x0028	GICV_AHPPIR	Virtual Machine Aliased Highest Priority Pending Interrupt Register
0x002C	GICV_STATUSR	Virtual Machine Error Reporting Status Register
0x00D0 + (4 * n)	GICV_APR<n>	Virtual Machine Active Priorities Registers
0x00FC	GICV_IIDR	Virtual Machine CPU Interface Identification Register
0x1000	GICV_DIR	Virtual Machine Deactivate Interrupt Register

In the GIC Redistributor block:

Frame	Offset	Name	Description
RD_base	0x0000	GICR_CTLR	Redistributor Control Register
RD_base	0x0004	GICR_IIDR	Redistributor Implementer Identification Register
RD_base	0x0008	GICR_TYPER	Redistributor Type Register
RD_base	0x0010	GICR_STATUSR	Error Reporting Status Register
RD_base	0x0010	GICR_STATUSR	Error Reporting Status Register
RD_base	0x0014	GICR_WAKER	Redistributor Wake Register
RD_base	0x0018	GICR_MPAMIDR	Report maximum PARTID and PMG Register
RD_base	0x001C	GICR_PARTIDR	Set PARTID and PMG Register
RD_base	0x0040	GICR_SETLPIR	Set LPI Pending Register
RD_base	0x0048	GICR_CLRLPIR	Clear LPI Pending Register
RD_base	0x0070	GICR_PROPBASER	Redistributor Properties Base Address Register
RD_base	0x0078	GICR_PENDBASER	Redistributor LPI Pending Table Base Address Register
RD_base	0x00A0	GICR_INVLPPIR	Redistributor Invalidate LPI Register
RD_base	0x00B0	GICR_INVALLR	Redistributor Invalidate All Register
RD_base	0x00C0	GICR_SYNCR	Redistributor Synchronize Register
SGI_base	0x0080	GICR_IGROUPR0	Interrupt Group Register 0
SGI_base	0x0080 + (4 * n)	GICR_IGROUPR<n>E	Interrupt Group Registers
SGI_base	0x0100	GICR_ISENBALER0	Interrupt Set-Enable Register 0
SGI_base	0x0100 + (4 * n)	GICR_ISENBALER<n>E	Interrupt Set-Enable Registers
SGI_base	0x0180	GICR_ICENABLER0	Interrupt Clear-Enable Register 0
SGI_base	0x0180 + (4 * n)	GICR_ICENABLER<n>E	Interrupt Clear-Enable Registers
SGI_base	0x0200	GICR_ISPENDR0	Interrupt Set-Pending Register 0
SGI_base	0x0200 + (4 * n)	GICR_ISPENDR<n>E	Interrupt Set-Pending Registers
SGI_base	0x0280	GICR_ICPENDR0	Interrupt Clear-Pending Register 0
SGI_base	0x0280 + (4 * n)	GICR_ICPENDR<n>E	Interrupt Clear-Pending Registers

Frame	Offset	Name	Description
SGI_base	0x0300	GICR_ISACTIVER0	Interrupt Set-Active Register 0
SGI_base	0x0300 + (4 * n)	GICR_ISACTIVER<n>E	Interrupt Set-Active Registers
SGI_base	0x0380	GICR_ICACTIVER0	Interrupt Clear-Active Register 0
SGI_base	0x0380 + (4 * n)	GICR_ICACTIVER<n>E	Interrupt Clear-Active Registers
SGI_base	0x0400 + (4 * n)	GICR_IPRIORITYR<n>E	Interrupt Priority Registers (extended PPI range)
SGI_base	0x0400 + (4 * n)	GICR_IPRIORITYR<n>	Interrupt Priority Registers
SGI_base	0x0C00	GICR_ICFGR0	Interrupt Configuration Register 0
SGI_base	0x0C00 + (4 * n)	GICR_ICFGR<n>E	Interrupt configuration registers
SGI_base	0x0C04	GICR_ICFGR1	Interrupt Configuration Register 1
SGI_base	0x0D00	GICR_IGRPMDR0	Interrupt Group Modifier Register 0
SGI_base	0x0D00 + (4 * n)	GICR_IGRPMDR<n>E	Interrupt Group Modifier Registers
SGI_base	0x0E00	GICR_NSACR	Non-secure Access Control Register
SGI_base	0x0F80	GICR_INMIR0	Non-maskable Interrupt Register for PPIs.
SGI_base	0x0F80 + (4 * n)	GICR_INMIR<n>E	Non-maskable Interrupt Registers for Extended PPIs, x = 1 to 2.
VLPI_base	0x0070	GICR_VPROPBASER	Virtual Redistributor Properties Base Address Register
VLPI_base	0x0078	GICR_VPENDBASER	Virtual Redistributor LPI Pending Table Base Address Register
VLPI_base	0x0080	GICR_VSGIR	Redistributor virtual SGI pending state request register
VLPI_base	0x0088	GICR_VSGIPENDR	Redistributor virtual SGI pending state register

In the GIC Virtual CPU interface block:

Offset	Name	Description
0x0000	GICV_CTLR	Virtual Machine Control Register
0x0004	GICV_PMR	Virtual Machine Priority Mask Register
0x0008	GICV_BPR	Virtual Machine Binary Point Register
0x000C	GICV_IAR	Virtual Machine Interrupt Acknowledge Register
0x0010	GICV_EOIR	Virtual Machine End Of Interrupt Register
0x0014	GICV_RPR	Virtual Machine Running Priority Register
0x0018	GICV_HPPIR	Virtual Machine Highest Priority Pending Interrupt Register
0x001C	GICV_ABPR	Virtual Machine Aliased Binary Point Register
0x0020	GICV_AIAR	Virtual Machine Aliased Interrupt Acknowledge Register
0x0024	GICV_AEOIR	Virtual Machine Aliased End Of Interrupt Register
0x0028	GICV_AHPPIR	Virtual Machine Aliased Highest Priority Pending Interrupt Register
0x002C	GICV_STATUSR	Virtual Machine Error Reporting Status Register
0x00D0 + (4 * n)	GICV_APR<n>	Virtual Machine Active Priorities Registers
0x00FC	GICV_IIDR	Virtual Machine CPU Interface Identification Register
0x1000	GICV_DIR	Virtual Machine Deactivate Interrupt Register

In the CTI block:

Offset	Name	Description
0x000	CTICONTROL	CTI Control register
0x010	CTIINTACK	CTI Output Trigger Acknowledge register
0x014	CTIAPPSET	CTI Application Trigger Set register
0x018	CTIAPPCLEAR	CTI Application Trigger Clear register

Offset	Name	Description
0x01C	CTIAPPPULSE	CTI Application Pulse register
0x020 + (4 * n)	CTIINEN<n>	CTI Input Trigger to Output Channel Enable registers
0x0A0 + (4 * n)	CTIOUTEN<n>	CTI Input Channel to Output Trigger Enable registers
0x130	CTITRIGINSTATUS	CTI Trigger In Status register
0x134	CTITRIGOUTSTATUS	CTI Trigger Out Status register
0x138	CTICHINSTATUS	CTI Channel In Status register
0x13C	CTICHOUTSTATUS	CTI Channel Out Status register
0x140	CTIGATE	CTI Channel Gate Enable register
0x144	ASICCTL	CTI External Multiplexer Control register
0x150	CTIDEVCTL	CTI Device Control register
0xF00	CTIITCTRL	CTI Integration mode Control register
0xFA0	CTICLAIMSET	CTI CLAIM Tag Set register
0xFA4	CTICLAIMCLR	CTI CLAIM Tag Clear register
0xFA8	CTIDEVAFF0	CTI Device Affinity register 0
0xFAC	CTIDEVAFF1	CTI Device Affinity register 1
0xFB0	CTILAR	CTI Lock Access Register
0xFB4	CTILSR	CTI Lock Status Register
0xFB8	CTIAUTHSTATUS	CTI Authentication Status register
0xFBC	CTIDEVARCH	CTI Device Architecture register
0xFC0	CTIDEVID2	CTI Device ID register 2
0xFC4	CTIDEVID1	CTI Device ID register 1
0xFC8	CTIDEVID	CTI Device ID register 0
0xFCC	CTIDEVTYPE	CTI Device Type register
0xFD0	CTIPIDR4	CTI Peripheral Identification Register 4
0xFE0	CTIPIDR0	CTI Peripheral Identification Register 0
0xFE4	CTIPIDR1	CTI Peripheral Identification Register 1
0xFE8	CTIPIDR2	CTI Peripheral Identification Register 2
0xFEC	CTIPIDR3	CTI Peripheral Identification Register 3
0xFF0	CTICIDR0	CTI Component Identification Register 0
0xFF4	CTICIDR1	CTI Component Identification Register 1
0xFF8	CTICIDR2	CTI Component Identification Register 2
0xFFC	CTICIDR3	CTI Component Identification Register 3

In the GIC ITS control block:

Offset	Name	Description
0x0000	GITS_CTLR	ITS Control Register
0x0004	GITS_IIDR	ITS Identification Register
0x0008	GITS_TYPER	ITS Type Register
0x0010	GITS_MPAMIDR	Report maximum PARTID and PMG Register
0x0014	GITS_PARTIDR	Set PARTID and PMG Register
0x0018	GITS_MPIDR	Report ITS's affinity.
0x0040	GITS_STATUSR	ITS Error Reporting Status Register
0x0048	GITS_UMSIR	ITS Unmapped MSI register
0x0080	GITS_CBASER	ITS Command Queue Descriptor
0x0088	GITS_CWRITER	ITS Write Register
0x0090	GITS_CREADR	ITS Read Register
0x0100 + (8 * n)	GITS_BASER<n>	ITS Translation Table Descriptors
0x20020	GITS_SGIR	ITS SGI Register

In the GIC CPU interface block:

Offset	Name	Description
0x0000	GICC_CTLR	CPU Interface Control Register
0x0004	GICC_PMR	CPU Interface Priority Mask Register
0x0008	GICC_BPR	CPU Interface Binary Point Register
0x000C	GICC_IAR	CPU Interface Interrupt Acknowledge Register
0x0010	GICC_EOIR	CPU Interface End Of Interrupt Register
0x0014	GICC_RPR	CPU Interface Running Priority Register
0x0018	GICC_HPIR	CPU Interface Highest Priority Pending Interrupt Register
0x001C	GICC_ABPR	CPU Interface Aliased Binary Point Register
0x0020	GICC_AIAR	CPU Interface Aliased Interrupt Acknowledge Register
0x0024	GICC_AEOIR	CPU Interface Aliased End Of Interrupt Register
0x0028	GICC_AHPIR	CPU Interface Aliased Highest Priority Pending Interrupt Register
0x002C	GICC_STATUSR	CPU Interface Status Register
0x002C	GICC_STATUSR	CPU Interface Status Register
0x00D0 + (4 * n)	GICC_APR<n>	CPU Interface Active Priorities Registers
0x00E0 + (4 * n)	GICC_NSAPR<n>	CPU Interface Non-secure Active Priorities Registers
0x00FC	GICC_IIDR	CPU Interface Identification Register
0x1000	GICC_DIR	CPU Interface Deactivate Interrupt Register

In the Timer block:

Frame	Offset	Name	Description
CNTBaseN	0x000	CNTPCT[31:0]	Counter-timer Physical Count
CNTBaseN	0x004	CNTPCT[63:32]	Counter-timer Physical Count
CNTBaseN	0x008	CNTVCT[31:0]	Counter-timer Virtual Count
CNTBaseN	0x00C	CNTVCT[63:32]	Counter-timer Virtual Count
CNTBaseN	0x010	CNTFRQ	Counter-timer Frequency
CNTBaseN	0x014	CNTEL0ACR	Counter-timer EL0 Access Control Register
CNTBaseN	0x018	CNTVOFF[31:0]	Counter-timer Virtual Offset
CNTBaseN	0x01C	CNTVOFF[63:32]	Counter-timer Virtual Offset
CNTBaseN	0x020	CNTP_CVAL[31:0]	Counter-timer Physical Timer CompareValue
CNTBaseN	0x024	CNTP_CVAL[63:32]	Counter-timer Physical Timer CompareValue
CNTBaseN	0x028	CNTP_TVAL	Counter-timer Physical Timer TimerValue
CNTBaseN	0x02C	CNTP_CTL	Counter-timer Physical Timer Control
CNTBaseN	0x030	CNTV_CVAL[31:0]	Counter-timer Virtual Timer CompareValue
CNTBaseN	0x034	CNTV_CVAL[63:32]	Counter-timer Virtual Timer CompareValue
CNTBaseN	0x038	CNTV_TVAL	Counter-timer Virtual Timer TimerValue
CNTBaseN	0x03C	CNTV_CTL	Counter-timer Virtual Timer Control
CNTBaseN	0xFD0 + (4 * n)	CounterID<n>	Counter ID registers
CNTCTLBase	0x000	CNTFRQ	Counter-timer Frequency
CNTCTLBase	0x004	CNTNSAR	Counter-timer Non-secure Access Register
CNTCTLBase	0x008	CNTTIDR	Counter-timer Timer ID Register
CNTCTLBase	0x040 + (4 * n)	CNTACR<n>	Counter-timer Access Control Registers

Frame	Offset	Name	Description
CNTCTLBase	$0x080 + (8 * n)$	CNTVOFF<n>[31:0]	Counter-timer Virtual Offsets
CNTCTLBase	$0x084 + (8 * n)$	CNTVOFF<n>[63:32]	Counter-timer Virtual Offsets
CNTCTLBase	$0xFD0 + (4 * n)$	CounterID<n>	Counter ID registers
CNTControlBase	0x000	CNTCR	Counter Control Register
CNTControlBase	0x004	CNTSR	Counter Status Register
CNTControlBase	0x008	CNTCVI[63:0]	Counter Count Value register
CNTControlBase	0x020	CNTFID0	Counter Frequency ID
CNTControlBase	$0x020 + (4 * n)$	CNTFID<n>	Counter Frequency IDs, n > 0
CNTControlBase	0x10	CNTSCR	Counter Scale Register
CNTControlBase	0x1C	CNTID	Counter Identification Register
CNTControlBase	$0xFD0 + (4 * n)$	CounterID<n>	Counter ID registers
CNTELOBaseN	0x000	CNTPCT[31:0]	Counter-timer Physical Count
CNTELOBaseN	0x004	CNTPCT[63:32]	Counter-timer Physical Count
CNTELOBaseN	0x008	CNTVCT[31:0]	Counter-timer Virtual Count
CNTELOBaseN	0x00C	CNTVCT[63:32]	Counter-timer Virtual Count
CNTELOBaseN	0x010	CNTFRQ	Counter-timer Frequency
CNTELOBaseN	0x020	CNTP_CVAL[31:0]	Counter-timer Physical Timer CompareValue
CNTELOBaseN	0x024	CNTP_CVAL[63:32]	Counter-timer Physical Timer CompareValue
CNTELOBaseN	0x028	CNTP_TVAL	Counter-timer Physical Timer TimerValue
CNTELOBaseN	0x02C	CNTP_CTL	Counter-timer Physical Timer Control
CNTELOBaseN	0x030	CNTV_CVAL[31:0]	Counter-timer Virtual Timer CompareValue
CNTELOBaseN	0x034	CNTV_CVAL[63:32]	Counter-timer Virtual Timer CompareValue
CNTELOBaseN	0x038	CNTV_TVAL	Counter-timer Virtual Timer TimerValue
CNTELOBaseN	0x03C	CNTV_CTL	Counter-timer Virtual Timer Control
CNTELOBaseN	$0xFD0 + (4 * n)$	CounterID<n>	Counter ID registers
CNTReadBase	0x000	CNTCVI[63:0]	Counter Count Value register
CNTReadBase	$0xFD0 + (4 * n)$	CounterID<n>	Counter ID registers

In the GIC ITS translation block:

Offset	Name	Description
0x0040	GITS_TRANSLATER	ITS Translation Register

In the AMU block:

Offset	Name	Description
$0x000 + (8 * n)$	AMEVCNTR0<n>[31:0]	Activity Monitors Event Counter Registers 0
$0x004 + (8 * n)$	AMEVCNTR0<n>[63:32]	Activity Monitors Event Counter Registers 0
$0x100 + (8 * n)$	AMEVCNTR1<n>[31:0]	Activity Monitors Event Counter Registers 1
$0x104 + (8 * n)$	AMEVCNTR1<n>[63:32]	Activity Monitors Event Counter Registers 1
$0x400 + (4 * n)$	AMEVTYPER0<n>	Activity Monitors Event Type Registers 0
$0x480 + (4 * n)$	AMEVTYPER1<n>	Activity Monitors Event Type Registers 1
0xC00	AMCNTENSET0	Activity Monitors Count Enable Set Register 0
0xC04	AMCNTENSET1	Activity Monitors Count Enable Set Register 1
0xC20	AMCNTENCLR0	Activity Monitors Count Enable Clear Register 0
0xC24	AMCNTENCLR1	Activity Monitors Count Enable Clear Register 1

Offset	Name	Description
0xCE0	AMCGCR	Activity Monitors Counter Group Configuration Register
0xE00	AMCFGR	Activity Monitors Configuration Register
0xE04	AMCR	Activity Monitors Control Register
0xE08	AMIIDR	Activity Monitors Implementation Identification Register
0xFA8	AMDEVAFF0	Activity Monitors Device Affinity Register 0
0xFAC	AMDEVAFF1	Activity Monitors Device Affinity Register 1
0xFBC	AMDEVARCH	Activity Monitors Device Architecture Register
0xFCC	AMDEVTYPE	Activity Monitors Device Type Register
0xFD0	AMPIDR4	Activity Monitors Peripheral Identification Register 4
0xFE0	AMPIDR0	Activity Monitors Peripheral Identification Register 0
0xFE4	AMPIDR1	Activity Monitors Peripheral Identification Register 1
0xFE8	AMPIDR2	Activity Monitors Peripheral Identification Register 2
0xFEC	AMPIDR3	Activity Monitors Peripheral Identification Register 3
0xFF0	AMCIDR0	Activity Monitors Component Identification Register 0
0xFF4	AMCIDR1	Activity Monitors Component Identification Register 1
0xFF8	AMCIDR2	Activity Monitors Component Identification Register 2
0xFFC	AMCIDR3	Activity Monitors Component Identification Register 3

In the ETE block:

Offset	Name	Description
0x004	TRCPRGCTLR	Programming Control Register
0x00C	TRCSTATR	Trace Status Register
0x010	TRCCONFIGR	Trace Configuration Register
0x018	TRCAUXCTLR	Auxiliary Control Register
0x020	TRCEVENTCTL0R	Event Control 0 Register
0x024	TRCEVENTCTL1R	Event Control 1 Register
0x028	TRCRSR	Resources Status Register
0x02C	TRCSTALLCTLR	Stall Control Register
0x030	TRCTSCTLR	Timestamp Control Register
0x034	TRCSYNCPR	Synchronization Period Register
0x038	TRCCCCTLR	Cycle Count Control Register
0x03C	TRCBBCTLR	Branch Broadcast Control Register
0x040	TRCTRACEIDR	Trace ID Register
0x044	TRCQCTLR	Q Element Control Register
0x080	TRCVICTLR	ViewInst Main Control Register
0x084	TRCVIIECTLR	ViewInst Include/Exclude Control Register
0x088	TRCVISSCTLR	ViewInst Start/Stop Control Register
0x08C	TRCVIPCSSCTLR	ViewInst Start/Stop PE Comparator Control Register
0x100 + (4 * n)	TRCSEQEVR<n>	Sequencer State Transition Control Register <n>;
0x118	TRCSEQRSTEV	Sequencer Reset Control Register
0x11C	TRCSEQSTR	Sequencer State Register
0x120 + (4 * n)	TRCEXTINSEL<n>	External Input Select Register <n>;
0x140 + (4 * n)	TRCCNTRLDVR<n>	Counter Reload Value Register <n>;
0x150 + (4 * n)	TRCCNTCTLR<n>	Counter Control Register <n>;
0x160 + (4 * n)	TRCCNTVR<n>	Counter Value Register <n>;
0x180	TRCIDR8	ID Register 8
0x184	TRCIDR9	ID Register 9
0x188	TRCIDR10	ID Register 10

Offset	Name	Description
0x18C	TRCIDR11	ID Register 11
0x190	TRCIDR12	ID Register 12
0x194	TRCIDR13	ID Register 13
0x1C0	TRCIMSPEC0	IMP DEF Register 0
0x1C0 + (4 * n)	TRCIMSPEC<n>	IMP DEF Register <n>;
0x1E0	TRCIDR0	ID Register 0
0x1E4	TRCIDR1	ID Register 1
0x1E8	TRCIDR2	ID Register 2
0x1EC	TRCIDR3	ID Register 3
0x1F0	TRCIDR4	ID Register 4
0x1F4	TRCIDR5	ID Register 5
0x1F8	TRCIDR6	ID Register 6
0x1FC	TRCIDR7	ID Register 7
0x200 + (4 * n)	TRCRSCTL<n>	Resource Selection Control Register <n>;
0x280 + (4 * n)	TRCSSCCR<n>	Single-shot Comparator Control Register <n>;
0x2A0 + (4 * n)	TRCSSCSR<n>	Single-shot Comparator Control Status Register <n>;
0x2C0 + (4 * n)	TRCSSPICR<n>	Single-shot Processing Element Comparator Input Control Register <n>;
0x304	TRCOSLSR	Trace OS Lock Status Register
0x310	TRCPDCR	PowerDown Control Register
0x314	TRCPDSR	PowerDown Status Register
0x400 + (8 * n)	TRCACVR<n>	Address Comparator Value Register <n>;
0x480 + (8 * n)	TRCACATR<n>	Address Comparator Access Type Register <n>;
0x600 + (8 * n)	TRCCIDCVR<n>	Context Identifier Comparator Value Registers <n>;
0x640 + (8 * n)	TRCVMIDCVR<n>	Virtual Context Identifier Comparator Value Register <n>;
0x680	TRCCIDCCTLR0	Context Identifier Comparator Control Register 0
0x684	TRCCIDCCTLR1	Context Identifier Comparator Control Register 1
0x688	TRCVMIDCCTLR0	Virtual Context Identifier Comparator Control Register 0
0x68C	TRCVMIDCCTLR1	Virtual Context Identifier Comparator Control Register 1
0xF00	TRCITCTRL	Integration Mode Control Register
0xFA0	TRCCLAIMSET	Claim Tag Set Register
0xFA4	TRCCLAIMCLR	Claim Tag Clear Register
0xFA8	TRCDEVAFF	Device Affinity Register
0xFB0	TRCLAR	Lock Access Register
0xFB4	TRCLSR	Lock Status Register
0xFB8	TRCAUTHSTATUS	Authentication Status Register
0xFBC	TRCDEVARCH	Device Architecture Register
0xFC0	TRCDEVID2	Device Configuration Register 2
0xFC4	TRCDEVID1	Device Configuration Register 1
0xFC8	TRCDEVID	Device Configuration Register
0xFCC	TRCDEVTYPE	Device Type Register
0xFD0	TRCPIDR4	Peripheral Identification Register 4
0xFD4	TRCPIDR5	Peripheral Identification Register 5
0xFD8	TRCPIDR6	Peripheral Identification Register 6
0xFDC	TRCPIDR7	Peripheral Identification Register 7
0xFE0	TRCPIDR0	Peripheral Identification Register 0
0xFE4	TRCPIDR1	Peripheral Identification Register 1
0xFE8	TRCPIDR2	Peripheral Identification Register 2
0xFEC	TRCPIDR3	Peripheral Identification Register 3

Offset	Name	Description
0xFF0	TRCCIDR0	Component Identification Register 0
0xFF4	TRCCIDR1	Component Identification Register 1
0xFF8	TRCCIDR2	Component Identification Register 2
0xFFC	TRCCIDR3	Component Identification Register 3

In the MPAM block:

Frame	Offset	Name	Description
MPAMF_BASE_ns	0x0000	MPAMF_IDR	MPAM Features Identification Register
MPAMF_BASE_ns	0x0018	MPAMF_IIDR	MPAM Implementation Identification Register
MPAMF_BASE_ns	0x0020	MPAMF_AIDR	MPAM Architecture Identification Register
MPAMF_BASE_ns	0x0028	MPAMF_IMPL_IDR	MPAM Implementation-Specific Partitioning Feature Identification Register
MPAMF_BASE_ns	0x0030	MPAMF_CPOR_IDR	MPAM Features Cache Portion Partitioning ID register
MPAMF_BASE_ns	0x0038	MPAMF_CCAP_IDR	MPAM Features Cache Capacity Partitioning ID register
MPAMF_BASE_ns	0x0040	MPAMF_MBW_IDR	MPAM Memory Bandwidth Partitioning Identification Register
MPAMF_BASE_ns	0x0048	MPAMF_PRI_IDR	MPAM Priority Partitioning Identification Register
MPAMF_BASE_ns	0x0050	MPAMF_PARTID_NRW_IDR	MPAM PARTID Narrowing ID register
MPAMF_BASE_ns	0x0080	MPAMF_MSMON_IDR	MPAM Resource Monitoring Identification Register
MPAMF_BASE_ns	0x0088	MPAMF_CSUMON_IDR	MPAM Features Cache Storage Usage Monitoring ID register
MPAMF_BASE_ns	0x0090	MPAMF_MBWUMON_IDR	MPAM Features Memory Bandwidth Usage Monitoring ID register
MPAMF_BASE_ns	0x00DC	MPAMF_ERR_MSI_MPAM	MPAM Error MSI Write MPAM Information Register
MPAMF_BASE_ns	0x00E0	MPAMF_ERR_MSI_ADDR_L	MPAM Error MSI Low-part Address Register
MPAMF_BASE_ns	0x00E4	MPAMF_ERR_MSI_ADDR_H	MPAM Error MSI High-part Address Register
MPAMF_BASE_ns	0x00E8	MPAMF_ERR_MSI_DATA	MPAM Error MSI Data Register
MPAMF_BASE_ns	0x00EC	MPAMF_ERR_MSI_ATTR	MPAM Error MSI Write Attributes Register
MPAMF_BASE_ns	0x00F0	MPAMF_ECR	MPAM Error Control Register
MPAMF_BASE_ns	0x00F8	MPAMF_ESR	MPAM Error Status Register
MPAMF_BASE_ns	0x0100	MPAMCFG_PART_SEL	MPAM Partition Configuration Selection Register

Frame	Offset	Name	Description
MPAMF_BASE_ns	0x0108	MPAMCFG_CMAX	MPAM Cache Maximum Capacity Partition Configuration Register
MPAMF_BASE_ns	0x0110	MPAMCFG_CMIN	MPAM Cache Minimum Capacity Partition Configuration Register
MPAMF_BASE_ns	0x0118	MPAMCFG_CASSOC	MPAM Cache Maximum Associativity Partition Configuration Register
MPAMF_BASE_ns	0x0200	MPAMCFG_MBW_MIN	MPAM Memory Bandwidth Minimum Partition Configuration Register
MPAMF_BASE_ns	0x0208	MPAMCFG_MBW_MAX	MPAM Memory Bandwidth Maximum Partition Configuration Register
MPAMF_BASE_ns	0x0220	MPAMCFG_MBW_WINWD	MPAM Memory Bandwidth Partitioning Window Width Configuration Register
MPAMF_BASE_ns	0x0300	MPAMCFG_EN	MPAM Partition Configuration Enable Register
MPAMF_BASE_ns	0x0310	MPAMCFG_DIS	MPAM Partition Configuration Disable Register
MPAMF_BASE_ns	0x0320	MPAMCFG_EN_FLAGS	MPAM Partition Configuration Enable Flags Register
MPAMF_BASE_ns	0x0400	MPAMCFG_PRI	MPAM Priority Partition Configuration Register
MPAMF_BASE_ns	0x0500	MPAMCFG_MBW_PROP	MPAM Memory Bandwidth Proportional Stride Partition Configuration Register
MPAMF_BASE_ns	0x0600	MPAMCFG_INTPARTID	MPAM Internal PARTID Narrowing Configuration Register
MPAMF_BASE_ns	0x0800	MSMON_CFG_MON_SEL	MPAM Monitor Instance Selection Register
MPAMF_BASE_ns	0x0808	MSMON_CAPT_EVNT	MPAM Capture Event Generation Register
MPAMF_BASE_ns	0x0810	MSMON_CFG_CSU_FLT	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Filter Register
MPAMF_BASE_ns	0x0818	MSMON_CFG_CSU_CTL	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Control Register
MPAMF_BASE_ns	0x0820	MSMON_CFG_MBWU_FLT	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Filter Register
MPAMF_BASE_ns	0x0828	MSMON_CFG_MBWU_CTL	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Control Register
MPAMF_BASE_ns	0x0840	MSMON_CSU	MPAM Cache Storage Usage Monitor Register
MPAMF_BASE_ns	0x0848	MSMON_CSU_CAPTURE	MPAM Cache Storage Usage Monitor Capture Register

Frame	Offset	Name	Description
MPAMF_BASE_ns	0x0858	MSMON_CSU_OFSR	MPAM CSU Monitor Overflow Status Register
MPAMF_BASE_ns	0x0860	MSMON_MBWU	MPAM Memory Bandwidth Usage Monitor Register
MPAMF_BASE_ns	0x0868	MSMON_MBWU_CAPTURE	MPAM Memory Bandwidth Usage Monitor Capture Register
MPAMF_BASE_ns	0x0880	MSMON_MBWU_L	MPAM Long Memory Bandwidth Usage Monitor Register
MPAMF_BASE_ns	0x0890	MSMON_MBWU_L_CAPTURE	MPAM Long Memory Bandwidth Usage Monitor Capture Register
MPAMF_BASE_ns	0x0898	MSMON_MBWU_OFSR	MPAM MBWU Monitor Overflow Status Register
MPAMF_BASE_ns	0x08DC	MSMON_OFLOW_MSI_MPAM	MPAM Monitor Overflow MSI Write MPAM Information Register
MPAMF_BASE_ns	0x08E0	MSMON_OFLOW_MSI_ADDR_L	MPAM Monitor Overflow MSI Low-part Address Register
MPAMF_BASE_ns	0x08E4	MSMON_OFLOW_MSI_ADDR_H	MPAM Monitor Overflow MSI Write High-part Address Register
MPAMF_BASE_ns	0x08E8	MSMON_OFLOW_MSI_DATA	MPAM Monitor Overflow MSI Write Data Register
MPAMF_BASE_ns	0x08EC	MSMON_OFLOW_MSI_ATTR	MPAM Monitor Overflow MSI Write Attributes Register
MPAMF_BASE_ns	0x08F0	MSMON_OFLOW_SR	MPAM Monitor Overflow Status Register
MPAMF_BASE_ns	0x1000 + (4 * n)	MPAMCFG_CPB<n>	MPAM Cache Portion Bitmap Partition Configuration Register
MPAMF_BASE_ns	0x2000 + (4 * n)	MPAMCFG_MBW_PBM<n>	MPAM Bandwidth Portion Bitmap Partition Configuration Register
MPAMF_BASE_rl	0x0000	MPAMF_IDR	MPAM Features Identification Register
MPAMF_BASE_rl	0x0018	MPAMF_IIDR	MPAM Implementation Identification Register
MPAMF_BASE_rl	0x0020	MPAMF_AIDR	MPAM Architecture Identification Register
MPAMF_BASE_rl	0x0028	MPAMF_IMPL_IDR	MPAM Implementation-Specific Partitioning Feature Identification Register
MPAMF_BASE_rl	0x0030	MPAMF_CPOR_IDR	MPAM Features Cache Portion Partitioning ID register
MPAMF_BASE_rl	0x0038	MPAMF_CCAP_IDR	MPAM Features Cache Capacity Partitioning ID register
MPAMF_BASE_rl	0x0040	MPAMF_MBW_IDR	MPAM Memory Bandwidth Partitioning Identification Register
MPAMF_BASE_rl	0x0048	MPAMF_PRI_IDR	MPAM Priority Partitioning Identification Register
MPAMF_BASE_rl	0x0050	MPAMF_PARTID_NRW_IDR	MPAM PARTID Narrowing ID register

Frame	Offset	Name	Description
MPAMF_BASE_ri	0x0080	MPAMF_MSMON_IDR	MPAM Resource Monitoring Identification Register
MPAMF_BASE_ri	0x0088	MPAMF_CSUMON_IDR	MPAM Features Cache Storage Usage Monitoring ID register
MPAMF_BASE_ri	0x0090	MPAMF_MBWUMON_IDR	MPAM Features Memory Bandwidth Usage Monitoring ID register
MPAMF_BASE_ri	0x00DC	MPAMF_ERR_MSI_MPAM	MPAM Error MSI Write MPAM Information Register
MPAMF_BASE_ri	0x00E0	MPAMF_ERR_MSI_ADDR_L	MPAM Error MSI Low-part Address Register
MPAMF_BASE_ri	0x00E4	MPAMF_ERR_MSI_ADDR_H	MPAM Error MSI High-part Address Register
MPAMF_BASE_ri	0x00E8	MPAMF_ERR_MSI_DATA	MPAM Error MSI Data Register
MPAMF_BASE_ri	0x00EC	MPAMF_ERR_MSI_ATTR	MPAM Error MSI Write Attributes Register
MPAMF_BASE_ri	0x00F0	MPAMF_ECR	MPAM Error Control Register
MPAMF_BASE_ri	0x00F8	MPAMF_ESR	MPAM Error Status Register
MPAMF_BASE_ri	0x0100	MPAMCFG_PART_SEL	MPAM Partition Configuration Selection Register
MPAMF_BASE_ri	0x0108	MPAMCFG_CMAX	MPAM Cache Maximum Capacity Partition Configuration Register
MPAMF_BASE_ri	0x0110	MPAMCFG_CMIN	MPAM Cache Minimum Capacity Partition Configuration Register
MPAMF_BASE_ri	0x0118	MPAMCFG_CASSOC	MPAM Cache Maximum Associativity Partition Configuration Register
MPAMF_BASE_ri	0x0200	MPAMCFG_MBW_MIN	MPAM Memory Bandwidth Minimum Partition Configuration Register
MPAMF_BASE_ri	0x0208	MPAMCFG_MBW_MAX	MPAM Memory Bandwidth Maximum Partition Configuration Register
MPAMF_BASE_ri	0x0220	MPAMCFG_MBW_WINWD	MPAM Memory Bandwidth Partitioning Window Width Configuration Register
MPAMF_BASE_ri	0x0300	MPAMCFG_EN	MPAM Partition Configuration Enable Register
MPAMF_BASE_ri	0x0310	MPAMCFG_DIS	MPAM Partition Configuration Disable Register
MPAMF_BASE_ri	0x0320	MPAMCFG_EN_FLAGS	MPAM Partition Configuration Enable Flags Register
MPAMF_BASE_ri	0x0400	MPAMCFG_PRI	MPAM Priority Partition Configuration Register
MPAMF_BASE_ri	0x0500	MPAMCFG_MBW_PROP	MPAM Memory Bandwidth Proportional Stride Partition Configuration Register

Frame	Offset	Name	Description
MPAMF_BASE_ri	0x0600	MPAMCFG_INTPARTID	MPAM Internal PARTID Narrowing Configuration Register
MPAMF_BASE_ri	0x0800	MSMON_CFG_MON_SEL	MPAM Monitor Instance Selection Register
MPAMF_BASE_ri	0x0808	MSMON_CAPT_EVNT	MPAM Capture Event Generation Register
MPAMF_BASE_ri	0x0810	MSMON_CFG_CSU_FLT	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Filter Register
MPAMF_BASE_ri	0x0818	MSMON_CFG_CSU_CTL	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Control Register
MPAMF_BASE_ri	0x0820	MSMON_CFG_MBWU_FLT	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Filter Register
MPAMF_BASE_ri	0x0828	MSMON_CFG_MBWU_CTL	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Control Register
MPAMF_BASE_ri	0x0840	MSMON_CSU	MPAM Cache Storage Usage Monitor Register
MPAMF_BASE_ri	0x0848	MSMON_CSU_CAPTURE	MPAM Cache Storage Usage Monitor Capture Register
MPAMF_BASE_ri	0x0858	MSMON_CSU_OFSR	MPAM CSU Monitor Overflow Status Register
MPAMF_BASE_ri	0x0860	MSMON_MBWU	MPAM Memory Bandwidth Usage Monitor Register
MPAMF_BASE_ri	0x0868	MSMON_MBWU_CAPTURE	MPAM Memory Bandwidth Usage Monitor Capture Register
MPAMF_BASE_ri	0x0880	MSMON_MBWU_L	MPAM Long Memory Bandwidth Usage Monitor Register
MPAMF_BASE_ri	0x0890	MSMON_MBWU_L_CAPTURE	MPAM Long Memory Bandwidth Usage Monitor Capture Register
MPAMF_BASE_ri	0x0898	MSMON_MBWU_OFSR	MPAM MBWU Monitor Overflow Status Register
MPAMF_BASE_ri	0x08DC	MSMON_OFLOW_MSI_MPAM	MPAM Monitor Overflow MSI Write MPAM Information Register
MPAMF_BASE_ri	0x08E0	MSMON_OFLOW_MSI_ADDR_L	MPAM Monitor Overflow MSI Low-part Address Register
MPAMF_BASE_ri	0x08E4	MSMON_OFLOW_MSI_ADDR_H	MPAM Monitor Overflow MSI Write High-part Address Register
MPAMF_BASE_ri	0x08E8	MSMON_OFLOW_MSI_DATA	MPAM Monitor Overflow MSI Write Data Register
MPAMF_BASE_ri	0x08EC	MSMON_OFLOW_MSI_ATTR	MPAM Monitor Overflow MSI Write Attributes Register
MPAMF_BASE_ri	0x08F0	MSMON_OFLOW_SR	MPAM Monitor Overflow Status Register
MPAMF_BASE_ri	0x1000 + (4 * n)	MPAMCFG_CPB<n>	MPAM Cache Portion Bitmap Partition Configuration Register

Frame	Offset	Name	Description
MPAMF_BASE_rl	0x2000 + (4 * n)	MPAMCFG_MBW_PBM<n>	MPAM Bandwidth Portion Bitmap Partition Configuration Register
MPAMF_BASE_rt	0x0000	MPAMF_IDR	MPAM Features Identification Register
MPAMF_BASE_rt	0x0018	MPAMF_IIDR	MPAM Implementation Identification Register
MPAMF_BASE_rt	0x0020	MPAMF_AIDR	MPAM Architecture Identification Register
MPAMF_BASE_rt	0x0028	MPAMF_IMPL_IDR	MPAM Implementation-Specific Partitioning Feature Identification Register
MPAMF_BASE_rt	0x0030	MPAMF_CPOR_IDR	MPAM Features Cache Portion Partitioning ID register
MPAMF_BASE_rt	0x0038	MPAMF_CCAP_IDR	MPAM Features Cache Capacity Partitioning ID register
MPAMF_BASE_rt	0x0040	MPAMF_MBW_IDR	MPAM Memory Bandwidth Partitioning Identification Register
MPAMF_BASE_rt	0x0048	MPAMF_PRI_IDR	MPAM Priority Partitioning Identification Register
MPAMF_BASE_rt	0x0050	MPAMF_PARTID_NRW_IDR	MPAM PARTID Narrowing ID register
MPAMF_BASE_rt	0x0080	MPAMF_MSMON_IDR	MPAM Resource Monitoring Identification Register
MPAMF_BASE_rt	0x0088	MPAMF_CSUMON_IDR	MPAM Features Cache Storage Usage Monitoring ID register
MPAMF_BASE_rt	0x0090	MPAMF_MBWUMON_IDR	MPAM Features Memory Bandwidth Usage Monitoring ID register
MPAMF_BASE_rt	0x00DC	MPAMF_ERR_MSI_MPAM	MPAM Error MSI Write MPAM Information Register
MPAMF_BASE_rt	0x00E0	MPAMF_ERR_MSI_ADDR_L	MPAM Error MSI Low-part Address Register
MPAMF_BASE_rt	0x00E4	MPAMF_ERR_MSI_ADDR_H	MPAM Error MSI High-part Address Register
MPAMF_BASE_rt	0x00E8	MPAMF_ERR_MSI_DATA	MPAM Error MSI Data Register
MPAMF_BASE_rt	0x00EC	MPAMF_ERR_MSI_ATTR	MPAM Error MSI Write Attributes Register
MPAMF_BASE_rt	0x00F0	MPAMF_ECR	MPAM Error Control Register
MPAMF_BASE_rt	0x00F8	MPAMF_ESR	MPAM Error Status Register
MPAMF_BASE_rt	0x0100	MPAMCFG_PART_SEL	MPAM Partition Configuration Selection Register
MPAMF_BASE_rt	0x0108	MPAMCFG_CMAX	MPAM Cache Maximum Capacity Partition Configuration Register
MPAMF_BASE_rt	0x0110	MPAMCFG_CMIN	MPAM Cache Minimum Capacity Partition Configuration Register

Frame	Offset	Name	Description
MPAMF_BASE_rt	0x0118	MPAMCFG_CASSOC	MPAM Cache Maximum Associativity Partition Configuration Register
MPAMF_BASE_rt	0x0200	MPAMCFG_MBW_MIN	MPAM Memory Bandwidth Minimum Partition Configuration Register
MPAMF_BASE_rt	0x0208	MPAMCFG_MBW_MAX	MPAM Memory Bandwidth Maximum Partition Configuration Register
MPAMF_BASE_rt	0x0220	MPAMCFG_MBW_WINWD	MPAM Memory Bandwidth Partitioning Window Width Configuration Register
MPAMF_BASE_rt	0x0300	MPAMCFG_EN	MPAM Partition Configuration Enable Register
MPAMF_BASE_rt	0x0310	MPAMCFG_DIS	MPAM Partition Configuration Disable Register
MPAMF_BASE_rt	0x0320	MPAMCFG_EN_FLAGS	MPAM Partition Configuration Enable Flags Register
MPAMF_BASE_rt	0x0400	MPAMCFG_PRI	MPAM Priority Partition Configuration Register
MPAMF_BASE_rt	0x0500	MPAMCFG_MBW_PROP	MPAM Memory Bandwidth Proportional Stride Partition Configuration Register
MPAMF_BASE_rt	0x0600	MPAMCFG_INTPARTID	MPAM Internal PARTID Narrowing Configuration Register
MPAMF_BASE_rt	0x0800	MSMON_CFG_MON_SEL	MPAM Monitor Instance Selection Register
MPAMF_BASE_rt	0x0808	MSMON_CAPT_EVNT	MPAM Capture Event Generation Register
MPAMF_BASE_rt	0x0810	MSMON_CFG_CSU_FLT	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Filter Register
MPAMF_BASE_rt	0x0818	MSMON_CFG_CSU_CTL	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Control Register
MPAMF_BASE_rt	0x0820	MSMON_CFG_MBWU_FLT	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Filter Register
MPAMF_BASE_rt	0x0828	MSMON_CFG_MBWU_CTL	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Control Register
MPAMF_BASE_rt	0x0840	MSMON_CSU	MPAM Cache Storage Usage Monitor Register
MPAMF_BASE_rt	0x0848	MSMON_CSU_CAPTURE	MPAM Cache Storage Usage Monitor Capture Register
MPAMF_BASE_rt	0x0858	MSMON_CSU_OFSR	MPAM CSU Monitor Overflow Status Register
MPAMF_BASE_rt	0x0860	MSMON_MBWU	MPAM Memory Bandwidth Usage Monitor Register
MPAMF_BASE_rt	0x0868	MSMON_MBWU_CAPTURE	MPAM Memory Bandwidth Usage Monitor Capture Register

Frame	Offset	Name	Description
MPAMF_BASE_rt	0x0880	MSMON_MBWU_L	MPAM Long Memory Bandwidth Usage Monitor Register
MPAMF_BASE_rt	0x0890	MSMON_MBWU_L_CAPTURE	MPAM Long Memory Bandwidth Usage Monitor Capture Register
MPAMF_BASE_rt	0x0898	MSMON_MBWU_OFSR	MPAM MBWU Monitor Overflow Status Register
MPAMF_BASE_rt	0x08DC	MSMON_OFLOW_MSI_MPAM	MPAM Monitor Overflow MSI Write MPAM Information Register
MPAMF_BASE_rt	0x08E0	MSMON_OFLOW_MSI_ADDR_L	MPAM Monitor Overflow MSI Low-part Address Register
MPAMF_BASE_rt	0x08E4	MSMON_OFLOW_MSI_ADDR_H	MPAM Monitor Overflow MSI Write High-part Address Register
MPAMF_BASE_rt	0x08E8	MSMON_OFLOW_MSI_DATA	MPAM Monitor Overflow MSI Write Data Register
MPAMF_BASE_rt	0x08EC	MSMON_OFLOW_MSI_ATTR	MPAM Monitor Overflow MSI Write Attributes Register
MPAMF_BASE_rt	0x08F0	MSMON_OFLOW_SR	MPAM Monitor Overflow Status Register
MPAMF_BASE_rt	0x1000 + (4 * n)	MPAMCFG_CPB<n>	MPAM Cache Portion Bitmap Partition Configuration Register
MPAMF_BASE_rt	0x2000 + (4 * n)	MPAMCFG_MBW_PBM<n>	MPAM Bandwidth Portion Bitmap Partition Configuration Register
MPAMF_BASE_s	0x0000	MPAMF_IDR	MPAM Features Identification Register
MPAMF_BASE_s	0x0008	MPAMF_SIDR	MPAM Features Secure Identification Register
MPAMF_BASE_s	0x0018	MPAMF_IIDR	MPAM Implementation Identification Register
MPAMF_BASE_s	0x0020	MPAMF_AIDR	MPAM Architecture Identification Register
MPAMF_BASE_s	0x0028	MPAMF_IMPL_IDR	MPAM Implementation-Specific Partitioning Feature Identification Register
MPAMF_BASE_s	0x0030	MPAMF_CPOR_IDR	MPAM Features Cache Portion Partitioning ID register
MPAMF_BASE_s	0x0038	MPAMF_CCAP_IDR	MPAM Features Cache Capacity Partitioning ID register
MPAMF_BASE_s	0x0040	MPAMF_MBW_IDR	MPAM Memory Bandwidth Partitioning Identification Register
MPAMF_BASE_s	0x0048	MPAMF_PRI_IDR	MPAM Priority Partitioning Identification Register
MPAMF_BASE_s	0x0050	MPAMF_PARTID_NRW_IDR	MPAM PARTID Narrowing ID register
MPAMF_BASE_s	0x0080	MPAMF_MSMON_IDR	MPAM Resource Monitoring Identification Register
MPAMF_BASE_s	0x0088	MPAMF_CSUMON_IDR	MPAM Features Cache Storage Usage Monitoring ID register

Frame	Offset	Name	Description
MPAMF_BASE_s	0x0090	MPAMF_MBWUMON_IDR	MPAM Features Memory Bandwidth Usage Monitoring ID register
MPAMF_BASE_s	0x00DC	MPAMF_ERR_MSI_MPAM	MPAM Error MSI Write MPAM Information Register
MPAMF_BASE_s	0x00E0	MPAMF_ERR_MSI_ADDR_L	MPAM Error MSI Low-part Address Register
MPAMF_BASE_s	0x00E4	MPAMF_ERR_MSI_ADDR_H	MPAM Error MSI High-part Address Register
MPAMF_BASE_s	0x00E8	MPAMF_ERR_MSI_DATA	MPAM Error MSI Data Register
MPAMF_BASE_s	0x00EC	MPAMF_ERR_MSI_ATTR	MPAM Error MSI Write Attributes Register
MPAMF_BASE_s	0x00F0	MPAMF_ECR	MPAM Error Control Register
MPAMF_BASE_s	0x00F8	MPAMF_ESR	MPAM Error Status Register
MPAMF_BASE_s	0x0100	MPAMCFG_PART_SEL	MPAM Partition Configuration Selection Register
MPAMF_BASE_s	0x0108	MPAMCFG_CMAX	MPAM Cache Maximum Capacity Partition Configuration Register
MPAMF_BASE_s	0x0110	MPAMCFG_CMIN	MPAM Cache Minimum Capacity Partition Configuration Register
MPAMF_BASE_s	0x0118	MPAMCFG_CASSOC	MPAM Cache Maximum Associativity Partition Configuration Register
MPAMF_BASE_s	0x0200	MPAMCFG_MBW_MIN	MPAM Memory Bandwidth Minimum Partition Configuration Register
MPAMF_BASE_s	0x0208	MPAMCFG_MBW_MAX	MPAM Memory Bandwidth Maximum Partition Configuration Register
MPAMF_BASE_s	0x0220	MPAMCFG_MBW_WINWD	MPAM Memory Bandwidth Partitioning Window Width Configuration Register
MPAMF_BASE_s	0x0300	MPAMCFG_EN	MPAM Partition Configuration Enable Register
MPAMF_BASE_s	0x0310	MPAMCFG_DIS	MPAM Partition Configuration Disable Register
MPAMF_BASE_s	0x0320	MPAMCFG_EN_FLAGS	MPAM Partition Configuration Enable Flags Register
MPAMF_BASE_s	0x0400	MPAMCFG_PRI	MPAM Priority Partition Configuration Register
MPAMF_BASE_s	0x0500	MPAMCFG_MBW_PROP	MPAM Memory Bandwidth Proportional Stride Partition Configuration Register
MPAMF_BASE_s	0x0600	MPAMCFG_INTPARTID	MPAM Internal PARTID Narrowing Configuration Register
MPAMF_BASE_s	0x0800	MSMON_CFG_MON_SEL	MPAM Monitor Instance Selection Register
MPAMF_BASE_s	0x0808	MSMON_CAPT_EVNT	MPAM Capture Event Generation Register

Frame	Offset	Name	Description
MPAMF_BASE_s	0x0810	MSMON_CFG_CSU_FLT	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Filter Register
MPAMF_BASE_s	0x0818	MSMON_CFG_CSU_CTL	MPAM Memory System Monitor Configure Cache Storage Usage Monitor Control Register
MPAMF_BASE_s	0x0820	MSMON_CFG_MBWU_FLT	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Filter Register
MPAMF_BASE_s	0x0828	MSMON_CFG_MBWU_CTL	MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Control Register
MPAMF_BASE_s	0x0840	MSMON_CSU	MPAM Cache Storage Usage Monitor Register
MPAMF_BASE_s	0x0848	MSMON_CSU_CAPTURE	MPAM Cache Storage Usage Monitor Capture Register
MPAMF_BASE_s	0x0858	MSMON_CSU_OFSR	MPAM CSU Monitor Overflow Status Register
MPAMF_BASE_s	0x0860	MSMON_MBWU	MPAM Memory Bandwidth Usage Monitor Register
MPAMF_BASE_s	0x0868	MSMON_MBWU_CAPTURE	MPAM Memory Bandwidth Usage Monitor Capture Register
MPAMF_BASE_s	0x0880	MSMON_MBWU_L	MPAM Long Memory Bandwidth Usage Monitor Register
MPAMF_BASE_s	0x0890	MSMON_MBWU_L_CAPTURE	MPAM Long Memory Bandwidth Usage Monitor Capture Register
MPAMF_BASE_s	0x0898	MSMON_MBWU_OFSR	MPAM MBWU Monitor Overflow Status Register
MPAMF_BASE_s	0x08DC	MSMON_OFLOW_MSI_MPAM	MPAM Monitor Overflow MSI Write MPAM Information Register
MPAMF_BASE_s	0x08E0	MSMON_OFLOW_MSI_ADDR_L	MPAM Monitor Overflow MSI Low-part Address Register
MPAMF_BASE_s	0x08E4	MSMON_OFLOW_MSI_ADDR_H	MPAM Monitor Overflow MSI Write High-part Address Register
MPAMF_BASE_s	0x08E8	MSMON_OFLOW_MSI_DATA	MPAM Monitor Overflow MSI Write Data Register
MPAMF_BASE_s	0x08EC	MSMON_OFLOW_MSI_ATTR	MPAM Monitor Overflow MSI Write Attributes Register
MPAMF_BASE_s	0x08F0	MSMON_OFLOW_SR	MPAM Monitor Overflow Status Register
MPAMF_BASE_s	0x1000 + (4 * n)	MPAMCFG_CPBM<n>	MPAM Cache Portion Bitmap Partition Configuration Register
MPAMF_BASE_s	0x2000 + (4 * n)	MPAMCFG_MBW_PBM<n>	MPAM Bandwidth Portion Bitmap Partition Configuration Register

In the RAS block:

Offset	Name	Description
0x000 + (64 * n)	ERR<n>FR	Error Record Feature Register
0x008 + (64 * n)	ERR<n>CTLR	Error Record Control Register
0x010 + (64 * n)	ERR<n>STATUS	Error Record Primary Status Register
0x018 + (64 * n)	ERR<n>ADDR	Error Record Address Register
0x020 + (64 * n)	ERR<n>MISC0	Error Record Miscellaneous Register 0
0x028 + (64 * n)	ERR<n>MISC1	Error Record Miscellaneous Register 1
0x030 + (64 * n)	ERR<n>MISC2	Error Record Miscellaneous Register 2
0x038 + (64 * n)	ERR<n>MISC3	Error Record Miscellaneous Register 3
0x800 + (64 * n)	ERR<n>PFGF	Pseudo-fault Generation Feature Register
0x800 + (8 * n)	ERRIMPDEF<n>	IMPLEMENTATION DEFINED Register <n>
0x808 + (64 * n)	ERR<n>PFGCTL	Pseudo-fault Generation Control Register
0x810 + (64 * n)	ERR<n>PFGCDN	Pseudo-fault Generation Countdown Register
0xE00	ERRGSR	Error Group Status Register
0xE10	ERRIIDR	Implementation Identification Register
0xE80	ERRFHICR0	Fault Handling Interrupt Configuration Register 0
0xE80 + (8 * n)	ERRIRQCR<n>	Generic Error Interrupt Configuration Register
0xE88	ERRFHICR1	Fault Handling Interrupt Configuration Register 1
0xE8C	ERRFHICR2	Fault Handling Interrupt Configuration Register 2
0xE90	ERRERICR0	Error Recovery Interrupt Configuration Register 0
0xE98	ERRERICR1	Error Recovery Interrupt Configuration Register 1
0xE9C	ERRERICR2	Error Recovery Interrupt Configuration Register 2
0xEA0	ERRCRICR0	Critical Error Interrupt Configuration Register 0
0xEA8	ERRCRICR1	Critical Error Interrupt Configuration Register 1
0xEAC	ERRCRICR2	Critical Error Interrupt Configuration Register 2
0xEF8	ERRIRQSR	Error Interrupt Status Register
0xFA8	ERRDEVAFF	Device Affinity Register
0xFBC	ERRDEVARCH	Device Architecture Register
0xFC8	ERRDEVID	Device Configuration Register
0xFD0	ERRPIDR4	Peripheral Identification Register 4
0xFE0	ERRPIDR0	Peripheral Identification Register 0
0xFE4	ERRPIDR1	Peripheral Identification Register 1
0xFE8	ERRPIDR2	Peripheral Identification Register 2
0xFEC	ERRPIDR3	Peripheral Identification Register 3
0xFF0	ERRCIDR0	Component Identification Register 0
0xFF4	ERRCIDR1	Component Identification Register 1
0xFF8	ERRCIDR2	Component Identification Register 2
0xFFC	ERRCIDR3	Component Identification Register 3

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbf36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

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

RWVT, bit [4]

Read/write access to the Virtual Timer register [CNTV_CVAL](#), [CNTV_TVAL](#), and [CNTV_CTL](#), in frame <n>. **The possible values of this bit are:**

RWVT	Meaning
0b0	No access to the Virtual Timer registers in frame <n>. The registers are RES0.
0b1	Read/write access to the Virtual Timer registers in frame <n>.

The reset behavior of this field is:

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

RVOFF, bit [3]

Read-only access to [CNTVOFF](#), in frame <n>. **The possible values of this bit are:**

RVOFF	Meaning
0b0	No access to CNTVOFF in frame <n>. The register is RES0.
0b1	Read-only access to CNTVOFF in frame <n>.

The reset behavior of this field is:

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

RFRQ, bit [2]

Read-only access to [CNTFRQ](#), in frame <n>. **The possible values of this bit are:**

RFRQ	Meaning
0b0	No access to CNTFRQ in frame <n>. The register is RES0.
0b1	Read-only access to CNTFRQ in frame <n>.

The reset behavior of this field is:

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

RVCT, bit [1]

Read-only access to [CNTVCT](#), in frame <n>. **The possible values of this bit are:**

RVCT	Meaning
0b0	No access to CNTVCT in frame <n>. The register is RES0.
0b1	Read-only access to CNTVCT in frame <n>.

The reset behavior of this field is:

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

RPCT, bit [0]

Read-only access to [CNTPCT](#), in frame <n>. **The possible values of this bit are:**

RPCT	Meaning
0b0	No access to CNTPCT in frame <n>. The register is RES0.
0b1	Read-only access to CNTPCT in frame <n>.

The reset behavior of this field is:

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

(old)

htmldiff from-

(new)

CNTCV, Counter Count Value register

The CNTCV characteristics are:

Purpose

Indicates the current count value.

Configuration

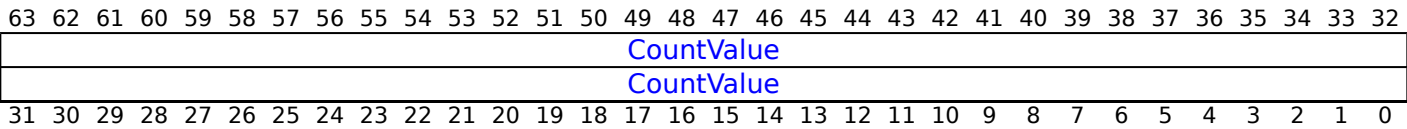
The power domain of CNTCV is IMPLEMENTATION DEFINED.

For more information, see 'Power and reset domains for the system level implementation of the Generic Timer'.

Attributes

CNTCV is a 64-bit register.

Field descriptions



CountValue, bits [63:0]

Indicates the counter value.

The reset behavior of this field is:

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

Accessing CNTCV

Frame	Accessibility
CNTControlBase	RW
CNTReadBase	RO

A write to CNTCV must be visible in the [CNTPCT](#) register of each running processor in a finite time.

For the instance of the register in the CNTControlBase frame:

- In a system that supports Secure and Non-secure memory maps, the CNTControlBase frame, and therefore this register instance, is implemented only in the Secure memory map.
- If the counter is enabled, the effect of writing to the register is UNKNOWN.

In an implementation that supports 64-bit atomic memory accesses, this register must be accessible using a 64-bit atomic access.

CNTCV can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance	Range
Timer	CNTControlBase	0x008	CNTCV	63:0

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance	Range
Timer	CNTReadBase	0x000	CNTCV	63:0

Accesses on this interface are **RO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTEL0ACR, Counter-timer EL0 Access Control Register

The CNTEL0ACR characteristics are:

Purpose

An implementation of CNTEL0ACR in the frame at CNTBaseN controls whether the [CNTPCT](#), [CNTVCT](#), [CNTFRQ](#), EL1 Physical Timer, and Virtual Timer registers are visible in the frame at CNTEL0BaseN.

Configuration

The power domain of CNTEL0ACR is IMPLEMENTATION DEFINED.

Implementation of this register is OPTIONAL.

For more information, see 'Power and reset domains for the system level implementation of the Generic Timer'.

Attributes

CNTEL0ACR 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										ELOPTEN		ELOVTEN		RES0				ELOVCTEN		ELOPCTEN											

Bits [31:10]

Reserved, RES0.

ELOPTEN, bit [9]

Second view read/write access control for the EL1 Physical Timer registers. This bit controls whether the [CNTP_CVAL](#), [CNTP_TVAL](#), and [CNTP_CTL](#) registers in the current CNTBaseN frame are also accessible in the corresponding CNTEL0BaseN frame. **The possible values of this bit are:**

ELOPTEN	Meaning
0b0	No access. Registers are RES0 in the second view.
0b1	Access permitted. If the registers are accessible in the current frame then they are accessible in the second view.

The reset behavior of this field is:

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

ELOVTEN, bit [8]

Second view read/write access control for the Virtual Timer registers. This bit controls whether the [CNTV_CVAL](#), [CNTV_TVAL](#), and [CNTV_CTL](#) registers in the current CNTBaseN frame are also accessible in the corresponding CNTEL0BaseN frame. **The possible values of this bit are:**

EL0VTEN	Meaning
0b0	No access. Registers are RES0 in the second view.
0b1	Access permitted. If the registers are accessible in the current frame then they are accessible in the second view.

The definition of this bit means that, if the Virtual Timer registers are not implemented in the current CNTBaseN frame, then the Virtual Timer register addresses are RES0 in the corresponding CNTEL0BaseN frame, regardless of the value of this bit.

The reset behavior of this field is:

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

Bits [7:2]

Reserved, RES0.

EL0VCTEN, bit [1]

Second view read access control for [CNTVCT](#) and [CNTERQ](#). The possible values of this bit are:

EL0VCTEN	Meaning
0b0	CNTVCT is not visible in the second view. If EL0PCTEN is set to 0, CNTERQ is not visible in the second view.
0b1	Access permitted. If CNTVCT and CNTERQ are visible in the current frame then they are visible in the second view.

The reset behavior of this field is:

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

EL0PCTEN, bit [0]

Second view read access control for [CNTPCT](#) and [CNTERQ](#). The possible values of this bit are:

EL0PCTEN	Meaning
0b0	CNTPCT is not visible in the second view. If EL0VCTEN is set to 0, CNTERQ is not visible in the second view.
0b1	Access permitted. If CNTPCT and CNTERQ are visible in the current frame then they are visible in the second view.

The reset behavior of this field is:

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

Accessing CNTEL0ACR

CNTEL0ACR can be implemented in any implemented CNTBaseN frame.

'CNTCTLBase status and control fields for the CNTBaseN and CNTEL0BaseN frames' describes the status fields that identify whether a CNTBaseN frame is implemented, and for an implemented frame:

- Whether the CNTBaseN frame has virtual timer capability.
- Whether the corresponding CNTEL0BaseN frame is implemented.
- For an implementation that recognizes two Security states, whether the CNTBaseN frame, and any corresponding CNTEL0BaseN frame, is accessible by Non-secure accesses.

If CNTEL0ACR is not implemented in an implemented CNTBaseN frame:

- The register location in that frame is RAZ/WI.

- If the corresponding CNTEL0BaseN frame is implemented, the registers [CNTERQ](#), [CNTP_CTL](#), [CNTP_CVAL](#), [CNTP_TVAL](#), [CNTPCT](#), [CNTV_CTL](#), [CNTV_CVAL](#), [CNTV_TVAL](#), and [CNTVCT](#) are not visible in that frame.

CNTEL0ACR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
Timer	CNTBaseN	0x014	CNTEL0ACR

Accesses on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

CNTNSAR, Counter-timer Non-secure Access Register

The CNTNSAR characteristics are:

Purpose

Provides the highest-level control of whether frames CNTBaseN and CNTELOBaseN are accessible by Non-secure accesses.

Configuration

The power domain of CNTNSAR is IMPLEMENTATION DEFINED.

For more information, see 'Power and reset domains for the system level implementation of the Generic Timer'.

Attributes

CNTNSAR 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																								NS7	NS6	NS5	NS4	NS3	NS2	NS1	NS0

Bits [31:8]

Reserved, RES0.

NS<n>, bit [n], for n = 7 to 0

Non-secure access to frame n. **The possible values of this bit are:**

NS<n>	Meaning
0b0	Secure access only. Behaves as RES0 to Non-secure accesses.
0b1	Secure and Non-secure accesses permitted.

This bit also determines whether, in the CNTCTLBase frame, [CNTACR<n>](#) and [CNTVOFF<n>](#) are accessible to Non-secure accesses.

If frame CNTBase<n>:

- Is not implemented, then NS<n> is RES0.
- Is not Configurable access, and is accessible only by Secure accesses, then NS<n> is RES0.
- Is not Configurable access, and is accessible by both Secure and Non-secure accesses, then NS<n> is RES1.

The reset behavior of this field is:

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

Accessing CNTNSAR

In a system that recognizes two Security states, this register is only accessible by Secure accesses.

CNTNSAR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
Timer	CNTCTLBase	0x004	CNTNSAR

Accesses on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

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

Bit [0]

Reserved, RES0.

Accessing CNTSR

In a system that supports Secure and Non-secure memory maps the CNTControlBase frame, that includes this register, is implemented only in the Secure memory map.

CNTSR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
Timer	CNTControlBase	0x004	CNTSR

Accesses on this interface are **RO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(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-A 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 This values is are: the same value as ID_AA64DFR0_EL1.DebugVer and DBGDIDR.Version. The defined values of this field are:

ARCHVER	Meaning
0b0110	Armv8.0 debug architecture.
0b0111	Armv8.0 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.

EDDEVARCH.ARCHVER and EDDEVARCH.ARCHPART are also defined as a single field, EDDEVARCH.ARCHID, so that EDDEVARCH.ARCHVER is EDDEVARCH.ARCHID[15:12].

FEAT_Debugv8p4 adds the functionality indicated by the value 0b1001. FEAT_Debugv8p2 adds the functionality indicated by the value 0b1000. If FEAT_VHE is not implemented, the only permitted value is 0b0110.

FEAT_VHE adds the functionality identified by the value 0b0111. The fields ARCHVER and ARCHPART together form the field ARCHID, so that ARCHVER is ARCHID[15:12].

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.

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.

ARCHPART, bits [11:0]

Architecture The Part. Defines number of the architecture Armv8-A of the debug component.

EDDEVARCH.ARCHVER and fields EDDEVARCH.ARCHPART ARCHVER are and also ARCHPART defined together as form a the single field, EDDEVARCH.ARCHID ARCHID, so that EDDEVARCH.ARCHPART ARCHPART is EDDEVARCH.ARCHID ARCHID[11:0].

Armv8-A debug architecture.

Reads as 0xA15.

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.

3020/09/2021 14:12:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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
RES0																				TraceFilt				UNKNOWN							
CTX_CMPs				RES0				WRPs				RES0				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 [63:44]

Reserved, RES0.

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]

Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints.

In an Armv8-A implementation that supports AArch64, this field returns the value of [ID_AA64DFR0_EL1](#).CTX_CMPs.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Bits [27:24]

Reserved, RES0.

WRPs, bits [23:20]

Number of watchpoints, minus 1. The value of 0b0000 is reserved.

In an Armv8-A implementation that supports AArch64, this field returns the value of [ID_AA64DFR0_EL1](#).WRPs.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Bits [19:16]

Reserved, RES0.

BRPs, bits [15:12]

Number of breakpoints, minus 1. The value of 0b0000 is reserved.

In an Armv8-A implementation that supports AArch64, this field returns the value of [ID_AA64DFR0_EL1](#).BRPs.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

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 also includes 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.control bit.
0b0101	PMUv3 for Armv8.4. As 0b0100, and adds also includes support for the PMMIR_EL1 register.
0b0110	PMUv3 for Armv8.5. As 0b0101, and adds also includes support for: <ul style="list-style-type: none"> 64-bit event counters. If EL2 is implemented, the MDCR_EL2.HCCD control.control bit. If EL3 is implemented, the MDCR_EL3.SCCD control.control bit.
0b0111	PMUv3 for Armv8.7. As 0b0110, and adds also includes support for: <ul style="list-style-type: none"> The PMCR_EL0.FZO and, if EL2 is implemented, MDCR_EL2.HPMFZO controls.control bits. If EL3 is implemented, the MDCR_EL3.{MPMX,MCCD} controls.control bits.
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.
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.

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.

TraceVer, bits [7:4]

Trace support. Indicates whether System register interface to a PE trace unit is implemented. Defined values are:

TraceVer	Meaning
0b0000	PE trace unit System registers not implemented.
0b0001	PE trace unit System registers implemented.

All other values are reserved.

A value of 0b0000 only indicates that no System register interface to a PE trace unit is implemented. A PE 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 **IMPEDEF**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

EDES, External Debug Event Status Register

The EDES characteristics are:

Purpose

Indicates the status of internally pending Halting debug events.

Configuration

EDES is in the Core power domain.

Attributes

EDES 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				ECSSSRRCOSUCOSUC														

Bits [31:43]

Reserved, RES0.

EC, bit [3]

When FEAT_Debugv8p8 is implemented:

Exception Catch debug event pending.

EC	Meaning
0b0	Exception Catch debug event is not pending.
0b1	Exception Catch debug event is pending.

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.

SS, bit [2]

When FEAT_DoPD is implemented:

Halting step debug event pending. Possible values of this field are:

SS	Meaning
0b0	Reading this means that a Halting step debug event is not pending. Writing this means no action.
0b1	Reading this means that a Halting step debug event is pending. Writing this clears the pending Halting step debug event.

The reset behavior of this field is:

- On a Cold reset, this field resets to 0.

Otherwise:

Halting step debug event pending. Possible values of this field are:

SS	Meaning
0b0	Reading this means that a Halting step debug event is not pending. Writing this means no action.
0b1	Reading this means that a Halting step debug event is pending. Writing this clears the pending Halting step debug event.

The reset behavior of this field is:

- On a Warm reset, this field resets to the value in [EDECR.SS](#).

RC, bit [1]

Reset Catch debug event pending. Possible values of this field are:

RC	Meaning
0b0	Reading this means that a Reset Catch debug event is not pending. Writing this means no action.
0b1	Reading this means that a Reset Catch debug event is pending. Writing this clears the pending Reset Catch debug event.

The reset behavior of this field is:

- On a Warm reset:
 - When FEAT_DoPD is implemented, this field resets to the value in [CTIDEVCTL.RCE](#).
 - When FEAT_DoPD is not implemented, this field resets to the value in [EDECR.RCE](#).

OSUC, bit [0]

OS Unlock Catch debug event pending. Possible values of this field are:

OSUC	Meaning
0b0	Reading this means that an OS Unlock Catch debug event is not pending. Writing this means no action.
0b1	Reading this means that an OS Unlock Catch debug event is pending. Writing this clears the pending OS Unlock Catch debug event.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

Accessing EDES

If a request to clear a pending Halting debug event is received at or about the time when halting becomes allowed, it is CONSTRAINED UNPREDICTABLE whether the event is taken.

If Core power is removed while a Halting debug event is pending, it is lost. However, it might become pending again when the Core is powered back on and Cold reset.

EDES can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0x020	EDES

This interface is accessible as follows:

- When IsCorePowered(), !DoubleLockStatus() and SoftwareLockStatus() accesses to this register are **RO**.
- When IsCorePowered(), !DoubleLockStatus() and !SoftwareLockStatus() accesses to this register are **RW**.
- Otherwise accesses to this register generate an error response.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

(old)

htmldiff from-

(new)

EDITR, External Debug Instruction Transfer Register

The EDITR characteristics are:

Purpose

Used in Debug state for passing instructions to the PE for execution.

Configuration

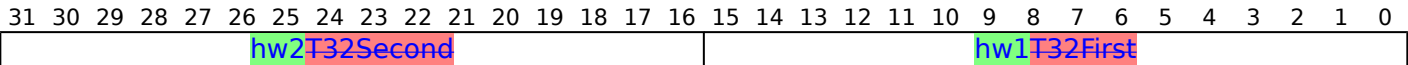
EDITR is in the Core power domain.

Attributes

EDITR is a 32-bit register.

Field descriptions

When AArch32 is supported and in AArch32 state:



hw2T32Second, bits [31:16]

Second halfword of the T32 instruction to be executed on the PE. When EDITR contains a 16-bit T32 instruction, this field is ignored. For more information, see 'Behavior in Debug state'.

Note

The hw2 field is displayed on the left. This is not the usual convention for display of T32 instruction halfwords.

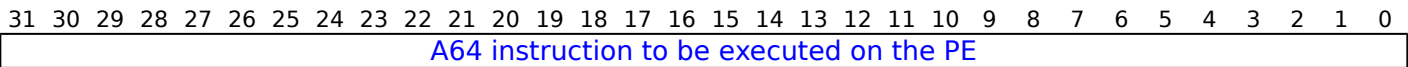
hw1T32First, bits [15:0]

First halfword of the T32 instruction to be executed on the PE.

Note

The hw1 field is displayed on the right. This is not the usual convention for display of T32 instruction halfwords.

When AArch64 is supported and in AArch64 state:



Bits [31:0]

A64 instruction to be executed on the PE.

(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 **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, **EDCDSR**, 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, **EDCDSR**, 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, **EDCDSR**, 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](#), 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, [EDCIDS](#), and [EDVIDSR](#). 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](#), and [EDVIDSR](#) 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 memory-mapped 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.

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ee9a85f5e501ea

Copyright © 2010-2021 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
TFORXfull		TXfull		ITORXO		TXUPipeAdv		ITEINTdis		TDAMASC2		NSRES0		SDDNSEHDE		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 [MDSCR_EL1](#) and [DBGDSCRext](#) System registers.

This bit is ignored by the PE when `any ExternalSecureNoninvasiveDebugEnabled() of == FALSE` and the following Effective is value true: `of MDCR_EL3.STE == 1`.

- `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:

If the PE is in Non-debug state, this bit is UNKNOWN. ITO is set to 0 on entry to Debug state.

Access to this field is **RO**.

- 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. the PE pipeline retires one or more instructions. Cleared to 0 by a write to **EDRCR.CSPA**.

The architecture does not define precisely when this bit is set to 1. It requires only that this happen periodically in Non-debug state to indicate 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 Memory Set and Copy instructions 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:

If the PE is in Non-debug state, this bit is UNKNOWN. It is always valid in Debug state.

Access to this field is **RO**.

- 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 [MDSCR_EL1](#) and [DBGDSCRext](#) System registers.

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 [MDSCR_EL1](#) and [DBGDSCRext](#) System registers.

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 [MDSCR_EL1](#) and [DBGDSCRext](#) System registers.

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>](#).

The possible values of this field are:

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	Secure.
0b0	0b1	Non-secure.
0b1	0b0	Root.
0b1	0b1	Realm.

Accessing this field has the following behavior:

In Non-debug state, this bit is UNKNOWN.

Access to this field is **RO**.

- 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~~When in Debug state, gives the current Security state:

NS	Meaning
0b0	Secure state.
0b1	Non-secure state.

Accessing this field has the following behavior:

~~In Non-debug state, this bit is UNKNOWN.~~

Access to this field is **RO**.

- 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:

Secure debug disabled.

Reports 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. The possible values of this field are:

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 In Non-debug state, this field has the following behavior: RAO.

Access to this field is RO.

- 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, this gives the current Exception level of the PE.

Accessing In Non-debug state, this field has the following behavior: RAZ.

Access to this field is RO.

- 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	SError 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:

UNKNOWN in Non-debug state.

Access to this field is **RO**.

- 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.

3020/09/2021 14:12:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

EDVIDSR, External Debug Virtual Context Sample Register

The EDVIDSR characteristics are:

Purpose

Contains sampled values captured on reading [EDPCSR](#)[31:0].

Configuration

EDVIDSR 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 EDVIDSR are RES0.

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.

When the PC Sample-based Profiling Extension is implemented in the external debug registers space, if EL2 is not implemented and EL3 is not implemented, it is IMPLEMENTATION DEFINED whether EDVIDSR is implemented.

Note

FEAT_PCSRv8p2 implements the PC Sample-based Profiling Extension in the Performance Monitors registers space.

Attributes

EDVIDSR is a 32-bit register.

Field descriptions

When FEAT_VHE is not implemented or EDSCR.SC2 == 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	E2	E3	HV	RES0												VMID [15:8]								VMID							

This format applies in all Armv8.0 implementations.

NS, bit [31]

Non-secure state sample. Indicates the Security state associated with the most recent [EDPCSR](#) 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.

E2, bit [30]**When EL2 is implemented:**

Exception level 2 status sample. Indicates whether the most recent [EDPCSR](#) sample was associated with EL2.

E2	Meaning
0b0	Sample is not from EL2.
0b1	Sample is from EL2.

The reset behavior of this field is:

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

Otherwise:

Reserved, RES0.

E3, bit [29]**When EL3 is implemented and AArch64 is supported:**

Exception level 3 status sample. Indicates whether the most recent [EDPCSR](#) sample was associated with EL3 using AArch64.

E3	Meaning
0b0	Sample is not from EL3 using AArch64.
0b1	Sample is from EL3 using AArch64.

The reset behavior of this field is:

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

Otherwise:

Reserved, RES0.

HV, bit [28]

EDPCSRhi ([EDPCSR](#)[63:32]) valid. Indicates whether bits [63:32] of the most recent [EDPCSR](#) sample might be nonzero:

HV	Meaning
0b0	Bits[63:32] of the most recent EDPCSR sample are zero.
0b1	Bits[63:32] of the most recent EDPCSR sample might be nonzero.

An EDVIDSR.HV value of 1 does not mean that the value of EDPCSRhi is nonzero. An EDVIDSR.HV value of 0 is a hint that EDPCSRhi ([EDPCSR](#)[63:32]) does not need to be read.

The reset behavior of this field is:

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

Bits [27:16]

Reserved, RES0.

VMID[15:8], bits [15:8]**When FEAT_VMID16 is implemented and EL2 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]**When EL2 is implemented:**

VMID sample. The VMID associated with the most recent EDPCSRlo ([EDPCSR](#)[31:0]) sample. When the most recent [EDPCSR](#) sample was generated:

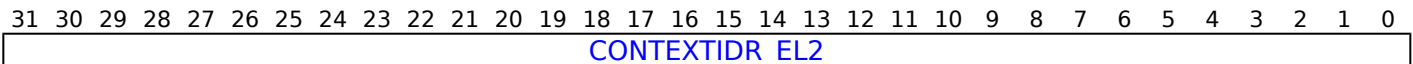
- This field is RES0 if any of the following apply:
 - The PE is executing in Secure state.
 - The PE is executing at EL2.
- Otherwise:
 - If EL2 is using AArch64 and either FEAT_VMID16 is not implemented or [VTCR_EL2](#).VS is 1, this field is set to [VTTBR_EL2](#).VMID.
 - If EL2 is using AArch64, 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.

The reset behavior of this field is:

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

Otherwise:

Reserved, RES0.

When (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented) and EDSCR.SC2 == 1:**CONTEXTIDR_EL2, bits [31:0]**

Context ID. The value of [CONTEXTIDR_EL2](#) that is associated with the most recent [EDPCSR](#) sample. When the most recent [EDPCSR](#) sample ~~is~~~~was~~ generated:

- If ~~the PE is not executing at EL3~~, EL2 ~~is~~~~was~~ using AArch64, and ~~EL2~~~~the~~ ~~is~~~~PE enabled~~~~was executing in the current SecurityNon-secure state~~, then this field is set to the Context ID sampled from [CONTEXTIDR_EL2](#).
- ~~Otherwise~~~~If EL2 was using AArch32 or the PE was executing in Secure state~~, ~~then~~ this field is set to an UNKNOWN value.

The reset behavior of this field is:

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

Accessing EDVIDSR

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'.

EDVIDSR can be accessed through the external debug interface:

Component	Offset	Instance
Debug	0x0A8	EDVIDSR

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.

3020/09/2021 14:53:09; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICD_CLRSPI_NSR, Clear Non-secure SPI Pending Register

The GICD_CLRSPI_NSR characteristics are:

Purpose

Removes the pending state from a valid SPI if permitted by the Security state of the access and the [GICD_NSACR<n>](#) value for that SPI.

A write to this register changes the state of a pending SPI to inactive, and the state of an active and pending SPI to active.

Configuration

If [GICD_TYPER](#).MBIS == 0, this register is reserved.

When [GICD_CTLR](#).DS == 1, this register provides functionality for all SPIs.

Attributes

GICD_CLRSPI_NSR 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																				INTID											

Bits [31:13]

Reserved, RES0.

INTID, bits [12:0]

The INTID of the SPI.

The function of this register depends on whether the targeted SPI is configured to be an edge-triggered or level-sensitive interrupt:

- For an edge-triggered interrupt, a write to [GICD_SETSPI_NSR](#) or [GICD_SETSPI_SR](#) adds the pending state to the targeted interrupt. It will stop being pending on activation, or if the pending state is removed by a write to [GICD_CLRSPI_NSR](#), [GICD_CLRSPI_SR](#), or [GICD_ICPENDR<n>](#).
- For a level-sensitive interrupt, a write to [GICD_SETSPI_NSR](#) or [GICD_SETSPI_SR](#) adds the pending state to the targeted interrupt. It will remain pending until it is deasserted by a write to [GICD_CLRSPI_NSR](#) or [GICD_CLRSPI_SR](#). If the interrupt is activated between having the pending state added and being deactivated, then the interrupt will be active and pending.

Accessing GICD_CLRSPI_NSR

Writes to this register have no effect if:

- The value written specifies a Secure SPI, the value is written by a Non-secure access, and the value of the corresponding [GICD_NSACR<n>](#) register is less than 0b10.
- The value written specifies an invalid SPI.
- The SPI is not pending.

A Secure access to this register can clear the pending state of any valid SPI.

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0048	GICD_CLRSPI_NSR

- AccessesThis oninterface thisis interfaceaccessible areas follows: **WO**.

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

(old)

htmldiff from-

(new)

GICD_CLRSPI_SR, Clear Secure SPI Pending Register

The GICD_CLRSPI_SR characteristics are:

Purpose

Removes the pending state from a valid SPI.

A write to this register changes the state of a pending SPI to inactive, and the state of an active and pending SPI to active.

Configuration

If [GICD_TYPER](#).MBIS == 0, this register is reserved.

When [GICD_CTLR](#).DS == 1, this register is WI.

Attributes

GICD_CLRSPI_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
RES0																INTID															

Bits [31:13]

Reserved, RES0.

INTID, bits [12:0]

The INTID of the SPI.

The function of this register depends on whether the targeted SPI is configured to be an edge-triggered or level-sensitive interrupt:

- For an edge-triggered interrupt, a write to [GICD_SETSPI_NSR](#) or [GICD_SETSPI_SR](#) adds the pending state to the targeted interrupt. It will stop being pending on activation, or if the pending state is removed by a write to [GICD_CLRSPI_NSR](#), [GICD_CLRSPI_SR](#), or [GICD_ICPENDR<n>](#).
- For a level-sensitive interrupt, a write to [GICD_SETSPI_NSR](#) or [GICD_SETSPI_SR](#) adds the pending state to the targeted interrupt. It will remain pending until it is deasserted by a write to [GICD_CLRSPI_NSR](#) or [GICD_CLRSPI_SR](#). If the interrupt is activated between having the pending state added and being deactivated, then the interrupt will be active and pending.

Accessing GICD_CLRSPI_SR

Writes to this register have no effect if:

- The value is written by a Non-secure access.
- The value written specifies an invalid SPI.
- The SPI is not pending.

16-bit accesses to bits [15:0] of this register must be supported.

This interface is accessible as follows:

- 3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

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

(old)

htmldiff from-

(new)

GICD_CPENDSGIR<n>, SGI Clear-Pending Registers, n = 0 - 3

The GICD_CPENDSGIR<n> characteristics are:

Purpose

Removes the pending state from an SGI.

A write to this register changes the state of a pending SGI to inactive, and the state of an active and pending SGI to active.

Configuration

Four SGI clear-pending registers are implemented. Each register contains eight clear-pending bits for each of four SGIs, for a total of 16 possible SGIs.

In multiprocessor implementations, each PE has a copy of these registers.

Attributes

GICD_CPENDSGIR<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
SGI_clear_pending_bits3								SGI_clear_pending_bits2								SGI_clear_pending_bits1								SGI_clear_pending_bits0							

SGI_clear_pending_bits<x>, bits [8x+7:8x], for x = 3 to 0

Removes the pending state from SGI number $4n + x$ for the PE corresponding to the bit number written to.

Reads and writes have the following behavior:

SGI_clear_pending_bits<x>	Meaning
0x00	If read, indicates that the SGI from the corresponding PE is not pending and is not active and pending.
0x01	If written, has no effect. If read, indicates that the SGI from the corresponding PE is pending or is active and pending. If written, removes the pending state from the SGI for the corresponding PE.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For SGI ID m , generated by processing element C writing to the corresponding [GICD_SGIR](#) field, where DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_CPENDSGIR<n> number is given by $n = m \text{ DIV } 4$.
- The offset of the required register is $(0xF10 + (4n))$.
- The offset of the required field within the register GICD_CPENDSGIR<n> is given by $m \text{ MOD } 4$.
- The required bit in the 8-bit SGI clear-pending field m is bit C .

Accessing GICD_CPENDSGIR<n>

These registers are used only when affinity routing is not enabled. When affinity routing is enabled, this register is RES0. An implementation is permitted to make the register RAZ/WI in this case.

A register bit that corresponds to an unimplemented SGI is RAZ/WI.

These registers are byte-accessible.

If the GIC implementation supports two Security states:

- A register bit that corresponds to a Group 0 interrupt is RAZ/WI to Non-secure accesses.
- Register bits corresponding to unimplemented PEs are RAZ/WI.

GICD_CPENDSGIR<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0F10 + (4 * n)	GICD_CPENDSGIR<n>

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses This on interface this is interface accessible are as follows: **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd5b36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The GICD_CTLR characteristics are:

Purpose

Enables interrupts and affinity routing.

Configuration

The format of this register depends on the Security state of the access and the number of Security states supported, which is specified by GICD_CTLR.DS.

Attributes

GICD_CTLR is a 32-bit register.

Field descriptions

When access is Secure, in a system that supports two Security states:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RWP																								E1NWFDS	ARE_NS	ARE_S	RES0	EnableGrp1S	EnableGrp1NS	EnableGrp1S	EnableGrp1NS

RWP, bit [31]

Register Write Pending. Read only. Indicates whether a register write is in progress or not:

RWP	Meaning
0b0	No register write in progress. The effects of previous register writes to the affected register fields are visible to all logical components of the GIC architecture, including the CPU interfaces.
0b1	Register write in progress. The effects of previous register writes to the affected register fields are not guaranteed to be visible to all logical components of the GIC architecture, including the CPU interfaces, as the effects of the changes are still being propagated.

This field tracks writes to:

- GICD_CTLR[2:0], the Group Enables, for transitions from 1 to 0 only.
- GICD_CTLR[7:4], the ARE bits, E1NWF bit and DS bit.
- GICD_ICENABLER<n>.

Updates to other register fields are not tracked by this field.

The reset behavior of this field is:

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

Bits [30:8]

Reserved, RES0.

E1NWF, bit [7]

Enable 1 of N Wakeup Functionality.

It is IMPLEMENTATION DEFINED whether this bit is programmable, or RAZ/WI.

If it is implemented, then it has the following behavior:

E1NWF	Meaning
0b0	A PE that is asleep cannot be picked for 1 of N interrupts.
0b1	A PE that is asleep can be picked for 1 of N interrupts as determined by IMPLEMENTATION DEFINED controls.

The reset behavior of this field is:

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

DS, bit [6]

Disable Security.

DS	Meaning
0b0	Non-secure accesses are not permitted to access and modify registers that control Group 0 interrupts.
0b1	Non-secure accesses are permitted to access and modify registers that control Group 0 interrupts.

If DS is written from 0 to 1 when GICD_CTLR.ARE_S == 1, then GICD_CTLR.ARE for the single Security state is RAO/WI.

If the Distributor only supports a single Security state, this bit is RAO/WI.

If the Distributor supports two Security states, it IMPLEMENTATION DEFINED whether this bit is programmable or implemented as RAZ/WI.

When this field is set to 1, all accesses to GICD_CTLR access the single Security state view, and all bits are accessible.

When set to 1, this field can only be cleared by a hardware reset.

Writing this bit from 0 to 1 is UNPREDICTABLE if any of the following is true:

- [GICD_CTLR.EnableGrp0](#)==1.
- [GICD_CTLR.EnableGrp1S](#)==1.
- [GICD_CTLR.EnableGrp1NS](#)==1.
- One or more INTID is in the Active or Active and Pending state.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

ARE_NS, bit [5]

Affinity Routing Enable, Non-secure state.

ARE_NS	Meaning
0b0	Affinity routing disabled for Non-secure state.
0b1	Affinity routing enabled for Non-secure state.

When affinity routing is enabled for the Secure state, this field is RAO/WI.

Changing the ARE_NS settings from 0 to 1 is UNPREDICTABLE except when GICD_CTLR.EnableGrp1 Non-secure == 0.

Changing the ARE_NS settings from 1 to 0 is UNPREDICTABLE.

If GICv2 backwards compatibility for Non-secure state is not implemented, this field is RAO/WI.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

ARE_S, bit [4]

Affinity Routing Enable, Secure state.

ARE_S	Meaning
0b0	Affinity routing disabled for Secure state.
0b1	Affinity routing enabled for Secure state.

Changing the ARE_S setting from 0 to 1 is UNPREDICTABLE except when all of the following apply:

- GICD_CTLR.EnableGrp0==0.
- GICD_CTLR.EnableGrp1S==0.
- GICD_CTLR.EnableGrp1NS==0.

Changing the ARE_S settings from 1 to 0 is UNPREDICTABLE.

If GICv2 backwards compatibility for Secure state is not implemented, this field is RAO/WI.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Bit [3]

Reserved, RES0.

EnableGrp1S, bit [2]

Enable Secure Group 1 interrupts.

EnableGrp1S	Meaning
0b0	Secure Group 1 interrupts are disabled.
0b1	Secure Group 1 interrupts are enabled.

If GICD_CTLR.ARE_S == 0, this field is RES0.

The reset behavior of this field is:

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

EnableGrp1NS, bit [1]

Enable Non-secure Group 1 interrupts.

EnableGrp1NS	Meaning
0b0	Non-secure Group 1 interrupts are disabled.
0b1	Non-secure Group 1 interrupts are enabled.

Note

This field also controls whether LPIs are forwarded to the PE.

The reset behavior of this field is:

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

EnableGrp0, bit [0]

Enable Group 0 interrupts.

EnableGrp0	Meaning
0b0	Group 0 interrupts are disabled.
0b1	Group 0 interrupts are enabled.

The reset behavior of this field is:

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

When access is Non-secure, in a system that supports two Security states:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
RWP					RES0															ARE NS					RES0					EnableGrp1A					EnableGrp1B				

RWP, bit [31]

This bit is a read-only alias of the Secure GICD_CTLR.RWP bit.

Bits [30:5]

Reserved, RES0.

ARE_NS, bit [4]

This bit is a read-write alias of the Secure GICD CTLR.ARE_NS bit.

If GICv2 backwards compatibility for Non-secure state is not implemented, this field is RAO/WI.

Bits [3:2]

Reserved, RES0.

EnableGrp1A, bit [1]

If `ARE_NS == 1`, then this bit is a read-write alias of the Secure GICD_CTLR.EnableGrp1NS bit.

If ARE NS == 0, then this bit is RES0.

EnableGrp1, bit [0]

If ARE_NS == 0, then this bit is a read-write alias of the Secure GICD CTLR.EnableGrp1NS bit.

If ARE NS == 1, then this bit is RES0.

When in a system that supports only a single Security 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
RWP	RES0										nASSGReq		E1NWFDS	RES0	ARE	RES0	EnableGrp1	EnableGrp0													

RWP, bit [31]

Register Write Pending. Read only. Indicates whether a register write is in progress or not:

RWP	Meaning
0b0	No register write in progress. The effects of previous register writes to the affected register fields are visible to all logical components of the GIC architecture, including the CPU interfaces.
0b1	Register write in progress. The effects of previous register writes to the affected register fields are not guaranteed to be visible to all logical components of the GIC architecture, including the CPU interfaces, as the effects of the changes are still being propagated.

This field tracks updates to:

- GICD_CTLR[2:0], the Group Enables, for transitions from 1 to 0 only.
- GICD_CTLR[7:4], the ARE bits, E1NWF bit and DS bit.
- GICD_ICENABLER<n>, the bits that allow disabling of SPIs.

Updates to other register fields are not tracked by this field.

The reset behavior of this field is:

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

Bits [30:9]

Reserved, RES0.

nASSGReq, bit [8]

When FEAT_GICv4p1 is implemented:

Controls whether SGIs have an active state.

This bit is RES0 if [GICD_TYPER2](#).GICD_TYPER2.nASSGReq is 0.

This bit is WI when any of GICD_CTLR.{EnableGrp0,EnableGrp1} is 1.

nASSGReq	Meaning
0b0	SGIs have an active state and must be deactivated.
0b1	SGIs do not have an active state and do not require deactivation.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Otherwise:

Reserved, RES0.

E1NWF, bit [7]

Enable 1 of N Wakeup Functionality.

It is IMPLEMENTATION DEFINED whether this bit is programmable, or RAZ/WI.

If it is implemented, then it has the following behavior:

E1NWF	Meaning
0b0	A PE that is asleep cannot be picked for 1 of N interrupts.
0b1	A PE that is asleep can be picked for 1 of N interrupts as determined by IMPLEMENTATION DEFINED controls.

The reset behavior of this field is:

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

DS, bit [6]

Disable Security. This field is RAO/WI.

Bit [5]

Reserved, RES0.

ARE, bit [4]

Affinity Routing Enable.

ARE	Meaning
0b0	Affinity routing disabled.
0b1	Affinity routing enabled.

Changing the ARE settings from 0 to 1 is UNPREDICTABLE except when all of the following apply:

- GICD_CTLR.EnableGrp1==0.
- GICD_CTLR.EnableGrp0==0.

Changing ARE from 1 to 0 is UNPREDICTABLE.

If GICv2 backwards compatibility is not implemented, this field is RAO/WI.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Bits [3:2]

Reserved, RES0.

EnableGrp1, bit [1]

Enable Group 1 interrupts.

EnableGrp1	Meaning
0b0	Group 1 interrupts disabled.
0b1	Group 1 interrupts enabled.

The reset behavior of this field is:

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

EnableGrp0, bit [0]

Enable Group 0 interrupts.

EnableGrp0	Meaning
0b0	Group 0 interrupts are disabled.
0b1	Group 0 interrupts are enabled.

The reset behavior of this field is:

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

Accessing GICD_CTLR

If an interrupt is pending within a CPU interface when the corresponding GICD_CTLR.EnableGrpX bit is written from 1 to 0 the interrupt must be retrieved from the CPU interface.

Note

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0000	GICD_CTLR

- AccessesThis oninterface thisis interfaceaccessible areas follows: RW.

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)	htmldiff from-	(new)

GICD_ICTIVER<n>, Interrupt Clear-Active Registers, n = 0 - 31

The GICD_ICTIVER<n> characteristics are:

Purpose

Deactivates the corresponding interrupt. These registers are used when saving and restoring GIC state.

Configuration

These registers are available in all GIC configurations. If GICD_CTLR.DS==0, these registers are Common.

The number of implemented GICD_ICTIVER<n> registers is (GICD_TYPER.ITLinesNumber+1). Registers are numbered from 0.

GICD_ICTIVER0 is Banked for each connected PE with GICR_TYPER.Processor_Number < 8.

Accessing GICD_ICTIVER0 from a PE with GICR_TYPER.Processor_Number > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

Attributes

GICD_ICTIVER<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	
Clear_active_bit31	Clear_active_bit30	Clear_active_bit29	Clear_active_bit28	Clear_active_bit27	Clear_active_bit26	Clear_active_bit25

Clear_active_bit<x>, bit [x], for x = 31 to 0

Removes the active state from interrupt number 32n + x. Reads and writes have the following behavior:

Clear_active_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not active, and is not active and pending. If written, has no effect.
0b1	If read, indicates that the corresponding interrupt is active, or is active and pending. If written, deactivates the corresponding interrupt, if the interrupt is active. If the interrupt is already deactivated, the write has no effect.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_ICTIVER<n> number, n, is given by n = m DIV 32.
- The offset of the required GICD_ICTIVER is (0x380 + (4*n)).
- The bit number of the required group modifier bit in this register is m MOD 32.

(old)

htmldiff from-

(new)

GICD_ICACTIVER<n>E, Interrupt Clear-Active Registers (extended SPI range), n = 0 - 31

The GICD_ICACTIVER<n>E characteristics are:

Purpose

Removes the active state from the corresponding SPI in the extended SPI range.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICD_ICACTIVER<n>E are RES0.

When GICD_TYPER.ESPI==0, these registers are RES0.

When GICD_TYPER.ESPI==1, the number of implemented GICD_ICACTIVER<n>E registers is (GICD_TYPER.ESPI_range+1). Registers are numbered from 0.

Attributes

GICD_ICACTIVER<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	26	Cle
Clear_active_bit31	Clear_active_bit30	Clear_active_bit29	Clear_active_bit28	Clear_active_bit27	Clear_active_bit26	Cle

Clear_active_bit<x>, bit [x], for x = 31 to 0

For the extended SPIs, removes the active state to interrupt number x. Reads and writes have the following behavior:

Clear active bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not active, and is not active and pending. If written, has no effect.
0b1	If read, indicates that the corresponding interrupt is active, or is active and pending. If written, deactivates the corresponding interrupt, if the interrupt is active. If the interrupt is already deactivated, the write has no effect.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_ICACTIVER<n>E number, n, is given by $n = (m-4096) \text{ DIV } 32$.
- The offset of the required GICD_ICACTIVER<n>E is $(0x1C00 + (4*n))$.
- The bit number of the required group modifier bit in this register is $(m-4096) \text{ MOD } 32$.

(old)

htmldiff from-

(new)

GICD_ICENABLER<n>, Interrupt Clear-Enable Registers, n = 0 - 31

The GICD_ICENABLER<n> characteristics are:

Purpose

Disables forwarding of the corresponding interrupt to the CPU interfaces.

Configuration

These registers are available in all GIC configurations. If [GICD_CTLR.DS](#)==0, these registers are Common.

The number of implemented [GICD_ICENABLER<n>](#) registers is ([GICD_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD_ICENABLER0 is Banked for each connected PE with [GICR_TYPER.Processor_Number](#) < 8.

Accessing GICD_ICENABLER0 from a PE with [GICR_TYPER.Processor_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

Attributes

GICD_ICENABLER<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26
Clear_enable_bit31	Clear_enable_bit30	Clear_enable_bit29	Clear_enable_bit28	Clear_enable_bit27	Clear_enable_bit26

Clear_enable_bit<x>, bit [x], for x = 31 to 0

For SPIs and PPIs, controls the forwarding of interrupt number 32n + x to the CPU interfaces. Reads and writes have the following behavior:

Clear_enable_bit<x>	Meaning
0b0	If read, indicates that forwarding of the corresponding interrupt is disabled. If written, has no effect.
0b1	If read, indicates that forwarding of the corresponding interrupt is enabled. If written, disables forwarding of the corresponding interrupt. After a write of 1 to this bit, a subsequent read of this bit returns 0.

For SGIs, the behavior of this bit is IMPLEMENTATION DEFINED.

The reset behavior of this field is:

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

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_ICENABLER<n> number, n, is given by n = m DIV 32.

Writing a 1 to a GICD_ICENABLER<n> bit only disables the forwarding of the corresponding interrupt from the Distributor to any CPU interface. It does not prevent the interrupt from changing state, for example becoming pending or active and pending if it is already active.

- When ARE is 1 for the Security state of an interrupt, the field for that interrupt is RES0 and an implementation is permitted to make the field RAZ/WI in this case.
- Equivalent functionality is provided by GICR ICENABLER0.

When `GICD_CTLR.DS==0`, bits corresponding to Group 0 and Secure Group 1 interrupts are RAZ/WI to Non-secure accesses.

It is IMPLEMENTATION DEFINED whether implemented SGIs are permanently enabled, or can be enabled and disabled by writes to [GICD_ISENABLER<n>](#) and [GICD_ICENABLER<n>](#) where n=0.

Completion of a write to this register does not guarantee that the effects of the write are visible throughout the affinity hierarchy. To ensure an enable has been cleared, software must write to the register with bits set to 1 to clear the required enables. Software must then poll **GICD_CTLR.RWP** until it has the value zero.

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0180 + (4 * n)	GICD_ICENABLER<n>

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RW**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbfdb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

(old)

htmldiff from-

(new)

GICD_ICENABLER<n>E, Interrupt Clear-Enable Registers, n = 0 - 31

The GICD_ICENABLER<n>E characteristics are:

Purpose

Disables forwarding of the corresponding SPI in the extended SPI range to the CPU interfaces.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICD_ICENABLER<n>E are RES0.

When [GICD_TYPER.ESPI](#)==0, these registers are RES0.

When [GICD_TYPER.ESPI](#)==1, the number of implemented [GICD_ICENABLER<n>E](#) registers is ([GICD_TYPER.ESPI_range](#)+1). Registers are numbered from 0.

Attributes

GICD_ICENABLER<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	26
Clear_enable_bit31	Clear_enable_bit30	Clear_enable_bit29	Clear_enable_bit28	Clear_enable_bit27	Clear_enable_bit26

Clear_enable_bit<x>, bit [x], for x = 31 to 0

For the extended SPI range, controls the forwarding of interrupt number x to the CPU interface. Reads and writes have the following behavior:

Clear enable bit<x>	Meaning
0b0	If read, indicates that forwarding of the corresponding interrupt is disabled. If written, has no effect.
0b1	If read, indicates that forwarding of the corresponding interrupt is enabled. If written, enables forwarding of the corresponding interrupt. After a write of 1 to this bit, a subsequent read of this bit returns 0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_ICENABLER<n>E number, n, is given by $n = (m-4096) \text{ DIV } 32$.
- The offset of the required GICD_ICENABLER<n>E is $(0 \times 1400 + (4 * n))$.
- The bit number of the required group modifier bit in this register is $(m-4096) \text{ MOD } 32$.

Accessing GICD_ICENABLER<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICD_ICENABLER<n>E, the corresponding bit is RES0.

When [GICD_CTLR.DS](#)==0, bits corresponding to Secure SPIs are RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICD_ICENABLER<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x1400 + (4 * n)	GICD_ICENABLER<n>E

- When [GICD_CTLR.DS](#) == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses on this interface are as follows: **RW**.

3020/09/2021 14:12:53; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICD_ICFGR<n>, Interrupt Configuration Registers, n = 0 - 63

The GICD_ICFGR<n> characteristics are:

Purpose

Determines whether the corresponding interrupt is edge-triggered or level-sensitive.

Configuration

These registers are available in all GIC configurations. If the GIC implementation supports two Security states, these registers are Common.

GICD_ICFGR1 is Banked for each connected PE with [GICR_TYPER](#).Processor_Number < 8.

Accessing GICD_ICFGR1 from a PE with [GICR_TYPER](#).Processor_Number > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

For SGIs and PPIs:

- When ARE is 1 for the Security state of an interrupt, the field for that interrupt is RES0 and an implementation is permitted to make the field RAZ/WI in this case.
- Equivalent functionality is provided by GICR_ICFGR<n>

For each supported PPI, it is IMPLEMENTATION DEFINED whether software can program the corresponding Int_config field.

For SGIs, Int_config fields are RO, meaning that GICD_ICFGR0 is RO.

Changing Int_config when the interrupt is individually enabled is UNPREDICTABLE.

Changing the interrupt configuration between level-sensitive and edge-triggered (in either direction) at a time when there is a pending interrupt will leave the interrupt in an UNKNOWN pending state.

Fields corresponding to unimplemented interrupts are RAZ/WI.

Attributes

GICD_ICFGR<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
Int_config15	Int_config14	Int_config13	Int_config12	Int_config11	Int_config10	Int_config9	Int_config8	Int_config7	Int_config6	Int_config5	Int_config4	Int_config3	Int_config2	Int_config1	Int_config0	Int_config0	Int_config0	Int_config0

Int_config<x>, bits [2x+1:2x], for x = 15 to 0

Indicates whether the interrupt is level-sensitive or edge-triggered.

Int_config<x>	Meaning
0b00	Corresponding interrupt is level-sensitive.
0b10	Corresponding interrupt is edge-triggered.

Int_config[0] (bit [2x]) is RES0.

For SGIs, this field always indicates edge-triggered.

The reset behavior of this field is:

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

Accessing GICD_ICFGR<n>

For SPIs and PPIs, when [GICD_CTLR.DS](#)=0, a register bit that corresponds to a Group 0 or Secure Group 1 interrupt is RAZ/WI to Non-secure accesses.

GICD_ICFGR<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0C00 + (4 * n)	GICD_ICFGR<n>

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: RW.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)htmldiff from-(new)

GICD_ICFGR<n>E, Interrupt Configuration Registers (Extended SPI Range), n = 0 - 63

The GICD_ICFGR<n>E characteristics are:

Purpose

Determines whether the corresponding SPI in the extended SPI range is edge-triggered or level-sensitive.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICD_ICFGR<n>E are RES0.

When GICD_TYPER.ESPI==0, these registers are RES0.

When GICD_TYPER.ESPI==1, the number of implemented GICD_ICFGR<n>E registers is ((GICD_TYPER.ESPI_range+1)*2). Registers are numbered from 0.

Attributes

GICD_ICFGR<n>E 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
Int_config15	Int_config14	Int_config13	Int_config12	Int_config11	Int_config10	Int_config9	Int_config8	Int_config7	Int_config6	Int_config5	Int_config4	Int_config3	Int_config2	Int_config1	Int_config0	Int_config0	Int_config0	Int_config0

Int_config<x>, bits [2x+1:2x], for x = 15 to 0

Indicates whether the interrupt is level-sensitive or edge-triggered.

Int_config[0] (bit[2x]) is RES0.

Int_config<x>	Meaning
0b00	Corresponding interrupt is level-sensitive.
0b10	Corresponding interrupt is edge-triggered.

The reset behavior of this field is:

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

Accessing GICD_ICFGR<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICD_ICFGR<n>E, the corresponding bit is RES0.

When GICD_CTLR.DS==0, a register bit that corresponds to a Group 0 or Secure Group 1 interrupt is RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICD_ICFGR<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
-----------	-------	--------	----------

GIC Distributor	Dist_base	0x3000 + (4 * n)	GICD_ICFGR<n>E
-----------------	-----------	------------------	----------------

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: RW.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fe44a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

GICD_ICPENDR<n>, Interrupt Clear-Pending Registers, n = 0 - 31

The GICD_ICPENDR<n> characteristics are:

Purpose

Removes the pending state from the corresponding interrupt.

Configuration

These registers are available in all GIC configurations. If [GICD_CTLR.DS](#)==0, these registers are Common.

The number of implemented [GICD_ICPENDR<n>](#) registers is ([GICD_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD_ICPENDR0 is Banked for each connected PE with [GICR_TYPER.Processor_Number](#) < 8.

Accessing GICD_ICPENDR0 from a PE with [GICR_TYPER.Processor_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

Attributes

GICD_ICPENDR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26
Clear_pending_bit31	Clear_pending_bit30	Clear_pending_bit29	Clear_pending_bit28	Clear_pending_bit27	Clear_pending_bit26

Clear_pending_bit<x>, bit [x], for x = 31 to 0

For SPIs and PPIs, removes the pending state from interrupt number 32n + x. Reads and writes have the following behavior:

Clear_pending_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not pending on any PE. If written, has no effect.
0b1	<ul style="list-style-type: none"> On this PE if the interrupt is an SGI or PPI. On at least one PE if the interrupt is an SPI. <p>If read, indicates that the corresponding interrupt is pending, or active and pending.</p> <p>If written, changes the state of the corresponding interrupt from pending to inactive, or from active and pending to active. This has no effect in the following cases:</p> <ul style="list-style-type: none"> If the interrupt is an SGI. In this case, the write is ignored. The pending state of an SGI can be cleared using GICD_CPENDSGIR<n>. If the interrupt is not pending and is not active and pending. If the interrupt is a level-sensitive interrupt that is pending or active and pending for a reason other than a write to GICD_ISPENDR<n>. In this case, if the interrupt signal continues to be asserted, the interrupt remains pending or active and pending.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_ICPENDR<n> number, n, is given by $n = m \text{ DIV } 32$.
- The offset of the required GICD_ICPENDR is $(0x200 + (4*n))$.
- The bit number of the required group modifier bit in this register is $m \text{ MOD } 32$.

Accessing GICD_ICPENDR<n>

Clear-pending bits for SGIs are RO/WI.

When affinity routing is enabled for the Security state of an interrupt:

- Bits corresponding to SGIs and PPIs are RAZ/WI, and equivalent functionality for SGIs and PPIs is provided by [GICR_ICPENDR0](#).
- Bits corresponding to Group 0 and Group 1 Secure interrupts can only be cleared by Secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

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 bits that correspond to Group 0 or Secure Group 1 interrupts are accessible only by Secure accesses and are RAZ/WI to Non-secure accesses.

GICD_ICPENDR<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0280 + (4 * n)	GICD_ICPENDR<n>

- When [GICD_CTLR.DS](#) == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses This on interface this is interface accessible areas follows: **RW**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

(old)

htmldiff from-

(new)

GICD_ICPENDR<n>E, Interrupt Clear-Pending Registers (extended SPI range), n = 0 - 31

The GICD_ICPENDR<n>E characteristics are:

Purpose

Removes the pending state to the corresponding SPI in the extended SPI range.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICD_ICPENDR<n>E are RES0.

When [GICD_TYPER](#).ESPI==0, these registers are RES0.

When [GICD_TYPER](#).ESPI==1, the number of implemented GICD_ICPENDR<n>E registers is ([GICD_TYPER](#).ESPI_range+1). Registers are numbered from 0.

Attributes

GICD_ICPENDR<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	26
Clear_pending_bit31	Clear_pending_bit30	Clear_pending_bit29	Clear_pending_bit28	Clear_pending_bit27	Clear_pending_bit26

Clear_pending_bit<x>, bit [x], for x = 31 to 0

For the extended PPIs, removes the pending state to interrupt number x. Reads and writes have the following behavior:

Clear_pending_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not pending. If written, has no effect.
0b1	If read, indicates that the corresponding interrupt is pending, or active and pending. If written, changes the state of the corresponding interrupt from pending to inactive, or from active and pending to active. This has no effect in the following cases: <ul style="list-style-type: none">If the interrupt is not pending and is not active and pending.If the interrupt is a level-sensitive interrupt that is pending or active and pending for a reason other than a write to GICD_ICPENDR<n>E. In this case, if the interrupt signal continues to be asserted, the interrupt remains pending or active and pending.

The reset behavior of this field is:

- The corresponding GICD_ICPENDR<n>E number, n, is given by $n = (m-4096) \text{ DIV } 32$.
- The offset of the required GICD_ICPENDR<n>E is $(0x1800 + (4*n))$.
- The bit number of the required group modifier bit in this register is $(m-4096) \text{ MOD } 32$.

(old)

htmldiff from-

(new)

GICD_IGROUPR<n>, Interrupt Group Registers, n = 0 - 31

The GICD_IGROUPR<n> characteristics are:

Purpose

Controls whether the corresponding interrupt is in Group 0 or Group 1.

Configuration

These registers are available in all GIC configurations. If [GICD_CTLR.DS](#)==0, these registers are Secure.

The number of implemented GICD_IGROUPR<n> registers is ([GICD_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD_IGROUPR0 is Banked for each connected PE with [GICR_TYPER.Processor_Number](#) < 8.

Accessing GICD_IGROUPR0 from a PE with [GICR_TYPER.Processor_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

Attributes

GICD_IGROUPR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26
Group_status_bit31	Group_status_bit30	Group_status_bit29	Group_status_bit28	Group_status_bit27	Group_status_bit26

Group_status_bit<x>, bit [x], for x = 31 to 0

Group status bit.

Group_status_bit<x>	Meaning
0b0	When GICD_CTLR.DS ==1, the corresponding interrupt is Group 0. When GICD_CTLR.DS ==0, the corresponding interrupt is Secure.
0b1	When GICD_CTLR.DS ==1, the corresponding interrupt is Group 1. When GICD_CTLR.DS ==0, the corresponding interrupt is Non-secure Group 1.

If affinity routing is enabled for the Security state of an interrupt, the bit that corresponds to the interrupt is concatenated with the equivalent bit in [GICD_IGRPMODR<n>](#) to form a 2-bit field that defines an interrupt group. The encoding of this field is described in [GICD_IGRPMODR<n>](#).

If affinity routing is disabled for the Security state of an interrupt, then:

- The corresponding [GICD_IGRPMODR<n>](#) bit is RES0.
- For Secure interrupts, the interrupt is Secure Group 0.
- For Non-secure interrupts, the interrupt is Non-secure Group 1.

- The corresponding GICD_IGROUPR<n>E number, n, is given by $n = (m-4096) \text{ DIV } 32$.
- The offset of the required GICD_IGROUPR<n>E is $(0 \times 1000 + (4 * n))$.
- The bit number of the required group modifier bit in this register is $(m-4096) \text{ MOD } 32$.

Accessing GICD_IGROUPR<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICD_IGROUPR<n>E, the corresponding bit is RES0.

When `GICD_CTLR.DS==0`, bits corresponding to Secure SPIs are RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICD_IGROUPR<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x1000 + (4 * n)	GICD_IGROUPR<n>E

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RW**.

30/09/2021 14:52:37; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICD_IGRPMDR<n>, Interrupt Group Modifier Registers, n = 0 - 31

The GICD_IGRPMDR<n> characteristics are:

Purpose

When [GICD_CTLR.DS](#)==0, this register together with the [GICD_IGROUPR<n>](#) registers, controls whether the corresponding interrupt is in:

- Secure Group 0.
- Non-secure Group 1.
- Secure Group 1.

Configuration

When [GICD_CTLR.DS](#)==0, these registers are Secure.

The number of implemented [GICD_IGROUPR<n>](#) registers is ([GICD_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

When [GICD_CTLR.ARE_S](#)==0 or [GICD_CTLR.DS](#)==1, the GICD_IGRPMDR<n> registers are RES0. An implementation can make these registers RAZ/WI in this case.

Attributes

GICD_IGRPMDR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	
Group_modifier_bit31	Group_modifier_bit30	Group_modifier_bit29	Group_modifier_bit28	Group_modifier_bit27	Group...

Group_modifier_bit<x>, bit [x], for x = 31 to 0

Group modifier bit. When affinity routing is enabled for the Security state of an interrupt, the bit that corresponds to the interrupt is concatenated with the equivalent bit in [GICD_IGROUPR<n>](#) to form a 2-bit field that defines an interrupt group:

Group modifier bit	Group status bit	Definition	Short name
0b0	0b0	Secure Group 0	G0S
0b0	0b1	Non-secure Group 1	G1NS
0b1	0b0	Secure Group 1	G1S
0b1	0b1	Reserved, treated as Non-secure Group 1	-

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_IGRPMDR<n> number, n, is given by $n = m \text{ DIV } 32$.
- The offset of the required GICD_IGRPMDR is $(0 \times 080 + (4 * n))$.
- The bit number of the required group modifier bit in this register is $m \text{ MOD } 32$.

See [GICD_IGROUPR<n>](#) for information about the GICD_IGRPMODR0 reset value.

Accessing GICD_IGRPMODR<n>

When affinity routing is enabled for Secure state, GICD_IGRPMODR0 is RES0 and equivalent functionality is proved by [GICR_IGRPMODR0](#).

When [GICD_CTLR](#).DS==0, the register is RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

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 interrupt is neither lost nor handled more than one time. The effect of the change must be visible in finite time.

GICD_IGRPMODR<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0D00 + (4 * n)	GICD_IGRPMODR<n>

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.

Accesses This on interface this is interface accessible areas follows: **RW**.

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICD_IGRPMODR<n>E, Interrupt Group Modifier Registers (extended SPI range), n = 0 - 31

The GICD_IGRPMODR<n>E characteristics are:

Purpose

When [GICD_CTLR.DS](#)==0, this register together with the [GICD_IGROUPR<n>E](#) registers, controls whether the corresponding interrupt is in:

- Secure Group 0.
- Non-secure Group 1.
- When System register access is enabled, Secure Group 1.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICD_IGRPMODR<n>E are RES0.

GICD_IGRPMODR<n>E resets to 0x00000000.

When [GICD_TYPER.ESPI](#)==0, these registers are RES0.

When [GICD_TYPER.ESPI](#)==1:

- The number of implemented GICD_IGRPMODR<n>E registers is ([GICD_TYPER.ESPI_range](#)+1). Registers are numbered from 0.
- When [GICD_CTLR.DS](#)==0, this register is Secure.

Attributes

GICD_IGRPMODR<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	
Group_modifier_bit31	Group_modifier_bit30	Group_modifier_bit29	Group_modifier_bit28	Group_modifier_bit27	Group_modifier_bit26

Group_modifier_bit<x>, bit [x], for x = 31 to 0

Group modifier bit. In implementations where affinity routing is enabled for the Security state of an interrupt, the bit that corresponds to the interrupt is concatenated with the equivalent bit in [GICD_IGROUPR<n>E](#) to form a 2-bit field that defines an interrupt group:

Group modifier bit	Group status bit	Definition	Short name
0b0	0b0	Secure Group 0	G0S
0b0	0b1	Non-secure Group 1	G1NS
0b1	0b0	Secure Group 1	G1S
0b1	0b1	Reserved, treated as Non-secure Group 1	-

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_IGRPMODR<n>E number, n, is given by $n = (m-4096) \text{ DIV } 32$.
- The offset of the required GICD_IGRPMODR<n>E is $(0x3400 + (4*n))$.
- The bit number of the required group modifier bit in this register is $(m-4096) \text{ MOD } 32$.

Accessing GICD_IGRPMODR<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICD_IGRPMODR<n>E, the corresponding bit is RES0.

When [GICD_CTLR.DS](#)==0, bits corresponding to Secure SPIs are RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICD_IGRPMODR<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	$0x3400 + (4 * n)$	GICD_IGRPMODR<n>E

- When [GICD_CTLR.DS](#) == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses on this interface are as follows: **RW**.

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICD_IIDR, Distributor Implementer Identification Register

The GICD_IIDR characteristics are:

Purpose

Provides information about the implementer and revision of the Distributor.

Configuration

This register is available in all configurations of the GIC. If the GIC implementation supports two Security states, this register is Common.

Attributes

GICD_IIDR 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
ProductID								RES0				Variant				Revision				Implementer											

ProductID, bits [31:24]

Product Identifier.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Bits [23:20]

Reserved, RES0.

Variant, bits [19:16]

Variant number. Typically, this field is used to distinguish product variants, or major revisions of a product.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Revision, bits [15:12]

Revision number. Typically, this field is used to distinguish minor revisions of a product.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Implementer, bits [11:0]

Contains the JEP106 code of the company that implemented the Distributor:

- Bits [11:8] are the JEP106 continuation code of the implementer. For an Arm implementation, this field is 0x4.
- Bit [7] is always 0.
- Bits [6:0] are the JEP106 identity code of the implementer. For an Arm implementation, bits [7:0] are therefore 0x3B.

Accessing GICD_IIDR

GICD_IIDR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0008	GICD_IIDR

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

Accesses This on interface this is interface accessible are as follows: **RO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

GICD_INMIR<n>, Non-maskable Interrupt Registers, x = 0 to 31, n = 0 - 31

The GICD_INMIR<n> characteristics are:

Purpose

Holds whether the corresponding SPI has the non-maskable property.

Configuration

This register is present only when FEAT_GICv3_NMI is implemented. Otherwise, direct accesses to GICD_INMIR<n> are RES0.

When [GICR_TYPER](#).NMI is 0, this register is RES0.

The number of implemented GICD_INMIR<n> registers is ([GICD_TYPER](#).ITLinesNumber+1). Registers are numbered from 0.

Attributes

GICD_INMIR<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
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 the non-maskable property.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_INMI<n> number, n, is given by $n = (m \text{ DIV } 32)$.
- The offset of the required GICD_INMI is $(0xF80 + (4*n))$.
- The bit number of the required in this register is $(m \text{ MOD } 32)$.

Accessing GICD_INMIR<n>

For SGIs and PPIs:

- The field for that interrupt is RES0 and an implementation is permitted to make the field RAZ/WI in this case.
- Equivalent functionality is provided by GICR_INMIR0.

When affinity routing is not enabled for the Security state of an interrupt in GICD_IGROUPR<n>E, the corresponding bit is RES0.

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICD_INMIR<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0F80 + (4 * n)	GICD_INMIR<n>

Accesses on this interface are **RW**.

30/09/2021 14:53; 092b4e1bbfbb45a293b198f9330c5f529ead2b0f

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

GICD_INMIR<n>E, Non-maskable Interrupt Registers for Extended SPIs, x = 0 to 31, n = 0 - 31

The GICD_INMIR<n>E characteristics are:

Purpose

Holds whether the corresponding SPI in the extended SPI range has the non-maskable property.

Configuration

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.

Attributes

GICD_INMIR<n>E 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 the non-maskable property.

If affinity routing is disabled for the Security state of an interrupt, the bit is RES0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_INMIR<n>E number, n, is given by $n = ((m-4096) \text{ DIV } 32)$.
- The offset of the required GICD_INMIR<n>E is $(0x3B00 + (4*n))$.
- The bit number in this register is $((m-4096) \text{ MOD } 32)$.

Accessing GICD_INMIR<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICD_IGROUPR<n>E, the corresponding bit is RES0.

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICD_INMIR<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x3B00 + (4 * n)	GICD_INMIR<n>E

Accesses on this interface are **RW**.

30/09/2021 14:53; 092b4e1bbfbb45a293b198f9330c5f529ead2b0f

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICD_IPRIORITYR<n>, Interrupt Priority Registers, n = 0 - 254

The GICD_IPRIORITYR<n> characteristics are:

Purpose

Holds the priority of the corresponding interrupt.

Configuration

These registers are available in all configurations of the GIC. When [GICD_CTLR.DS](#)=0, these registers are Common.

The number of implemented GICD_IPRIORITYR<n> registers is 8*([GICD_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD_IPRIORITYR0 to GICD_IPRIORITYR7 are Banked for each connected PE with [GICR_TYPER.Processor_Number](#) < 8.

Accessing GICD_IPRIORITYR0 to GICD_IPRIORITYR7 from a PE with [GICR_TYPER.Processor_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

Attributes

GICD_IPRIORITYR<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
Priority_offset_3B								Priority_offset_2B								Priority_offset_1B								Priority_offset_0B							

Priority_offset_3B, bits [31:24]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 3. Lower priority values correspond to greater priority of the interrupt.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Priority_offset_2B, bits [23:16]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 2. Lower priority values correspond to greater priority of the interrupt.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Priority_offset_1B, bits [15:8]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 1. Lower priority values correspond to greater priority of the interrupt.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Priority_offset_0B, bits [7:0]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 0. Lower priority values correspond to greater priority of the interrupt.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For interrupt ID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_IPRIORITYR<n> number, n, is given by $n = m \text{ DIV } 4$.
- The offset of the required GICD_IPRIORITYR<n> register is $(0x400 + (4 * n))$.
- The byte offset of the required Priority field in this register is $m \text{ MOD } 4$, where:
 - Byte offset 0 refers to register bits [7:0].
 - Byte offset 1 refers to register bits [15:8].
 - Byte offset 2 refers to register bits [23:16].
 - Byte offset 3 refers to register bits [31:24].

Accessing GICD_IPRIORITYR<n>

These registers are always used when affinity routing is not enabled. When affinity routing is enabled for the Security state of an interrupt:

- [GICR_IPRIORITYR<n>](#) is used instead of GICD_IPRIORITYR<n> where n = 0 to 7 (that is, for SGIs and PPIs).
- GICD_IPRIORITYR<n> is RAZ/WI where n = 0 to 7.

These registers are byte-accessible.

A register field corresponding to an unimplemented interrupt is RAZ/WI.

A GIC might implement fewer than eight priority bits, but must implement at least bits [7:4] of each field. In each field, unimplemented bits are RAZ/WI, see 'Interrupt prioritization' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

When [GICD_CTLR.DS](#)==0:

- A register bit that corresponds to a Group 0 or Secure Group 1 interrupt is RAZ/WI to Non-secure accesses.
- A Non-secure access to a field that corresponds to a Non-secure Group 1 interrupt behaves as described in 'Software accesses of interrupt priority' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

It is IMPLEMENTATION DEFINED whether changing the value of a priority field changes the priority of an active interrupt.

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 interrupt is neither lost nor handled more than one time. The effect of the change must be visible in finite time.

GICD_IPRIORITYR<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0400 + (4 * n)	GICD_IPRIORITYR<n>

- When [GICD_CTLR.DS](#) == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses This on interface this is interface accessible areas follows: RW.

3020/09/2021 1412:5237: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

- 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

In implementations that do not require 1 of N distribution of SPIs, this bit might be RAZ/WI.

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 $0x6000 + 8n$.

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

Note

GICD_IROUTER<n> can be accessed through the memory-mapped interfaces:

(old)

htmldiff from-

(new)

GICD_IROUTER<n>E, Interrupt Routing Registers (Extended SPI Range), n = 0 - 1023

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

In implementations that do not require 1 of N distribution of SPIs, this bit might be RAZ/WI.

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 $0 \times 6000 + 8n$.

Accessing GICD_IROUTER<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICD_IROUTER<n>E, the register is RES0.

When [GICD_CTLR.DS](#)==0, a register that corresponds to a Group 0 or Secure Group 1 interrupt is RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

Page 1378

(old)

htmldiff from-

(new)

GICD_ISACTIVER<n>E, Interrupt Set-Active Registers (extended SPI range), n = 0 - 31

The GICD_ISACTIVER<n>E characteristics are:

Purpose

Adds the active state to the corresponding SPI in the extended SPI range.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICD_ISACTIVER<n>E are RES0.

When GICD_TYPER.ESPI==0, these registers are RES0.

When GICD_TYPER.ESPI==1, the number of implemented GICD_ISACTIVER<n>E registers is (GICD_TYPER.ESPI_range+1). Registers are numbered from 0.

Attributes

GICD_ISACTIVER<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25
Set_active_bit31	Set_active_bit30	Set_active_bit29	Set_active_bit28	Set_active_bit27	Set_active_bit26	Set_active_bit25

Set_active_bit<x>, bit [x], for x = 31 to 0

For the extended SPIs, adds the active state to interrupt number x. Reads and writes have the following behavior:

Set_active_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not active, and is not active and pending. If written, has no effect.
0b1	If read, indicates that the corresponding interrupt is active, or active and pending on this PE. If written, activates the corresponding interrupt, if the interrupt is not already active. If the interrupt is already active, the write has no effect. After a write of 1 to this bit, a subsequent read of this bit returns 1.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_ISACTIVER<n>E number, n, is given by $n = (m-4096) \text{ DIV } 32$.
- The offset of the required GICD_ISACTIVER<n>E is $(0x1A00 + (4*n))$.
- The bit number of the required group modifier bit in this register is $(m-4096) \text{ MOD } 32$.

- The offset of the required GICD_ISENBLER is $(0x100 + (4*n))$.
- The bit number of the required group modifier bit in this register is $m \text{ MOD } 32$.

At start-up, and after a reset, a PE can use this register to discover which peripheral INTIDs the GIC supports. If [GICD_CTLR.DS](#)=0 in a system that supports EL3, the PE must do this for the Secure view of the available interrupts, and Non-secure software running on the PE must do this discovery after the Secure software has configured interrupts as Group 0/Secure Group 1 and Non-secure Group 1.

Accessing GICD_ISENABLER<n>

For SGIs and PPIs:

- When ARE is 1 for the Security state of an interrupt, the field for that interrupt is RES0 and an implementation is permitted to make the field RAZ/WI in this case.
- Equivalent functionality is provided by GICR_ISENABLER0.

Bits corresponding to unimplemented interrupts are RAZ/WI.

When `GICD_CTLR.DS==0`, bits corresponding to Group 0 or Secure Group 1 interrupts are RAZ/WI to Non-secure accesses.

It is IMPLEMENTATION DEFINED whether implemented SGIs are permanently enabled, or can be enabled and disabled by writes to [GICD_ISENABLER<n>](#) and [GICD_ICENABLER<n>](#) where n=0.

For SPIs and PPIs, each bit controls the forwarding of the corresponding interrupt from the Distributor to the CPU interfaces.

GICD_ISENBLER<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0100 + (4 * n)	GICD_ISENABLER<n>

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: RW.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

GICD_ISENABLER<n>E, Interrupt Set-Enable Registers, n = 0 - 31

The GICD_ISENABLER<n>E characteristics are:

Purpose

Enables forwarding of the corresponding SPI in the extended SPI range to the CPU interfaces.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICD_ISENABLER<n>E are RES0.

When [GICD_TYPER](#).ESPI==0, these registers are RES0.

When [GICD_TYPER](#).ESPI==1, the number of implemented [GICD_ISENABLER<n>E](#) registers is ([GICD_TYPER](#).ESPI_range+1). Registers are numbered from 0.

Attributes

GICD_ISENABLER<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25
Set_enable_bit31	Set_enable_bit30	Set_enable_bit29	Set_enable_bit28	Set_enable_bit27	Set_enable_bit26	Set_enable_bit25

Set_enable_bit<x>, bit [x], for x = 31 to 0

For the extended SPI range, controls the forwarding of interrupt number x to the CPU interface. Reads and writes have the following behavior:

Set_enable_bit<x>	Meaning
0b0	If read, indicates that forwarding of the corresponding interrupt is disabled. If written, has no effect.
0b1	If read, indicates that forwarding of the corresponding interrupt is enabled. If written, enables forwarding of the corresponding interrupt. After a write of 1 to this bit, a subsequent read of this bit returns 1.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_ISENABLER<n>E number, n, is given by $n = (m-4096) \text{ DIV } 32$.
- The offset of the required GICD_ISENABLER<n>E is $(0x1200 + (4*n))$.
- The bit number of the required group modifier bit in this register is $(m-4096) \text{ MOD } 32$.

(old)

htmldiff from-

(new)

GICD_ISPENDR<n>, Interrupt Set-Pending Registers, n = 0 - 31

The GICD_ISPENDR<n> characteristics are:

Purpose

Adds the pending state to the corresponding interrupt.

Configuration

These registers are available in all GIC configurations. If [GICD_CTLR.DS](#)==0, these registers are Common.

The number of implemented [GICD_ISPENDR<n>](#) registers is ([GICD_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD_ISPENDR0 is Banked for each connected PE with [GICR_TYPER.Processor_Number](#) < 8.

Accessing GICD_ISPENDR0 from a PE with [GICR_TYPER.Processor_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

Attributes

GICD_ISPENDR<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	
Set_pending_bit31	Set_pending_bit30	Set_pending_bit29	Set_pending_bit28	Set_pending_bit27	Set_pending_bit26	Set_pending_bit25

Set_pending_bit<x>, bit [x], for x = 31 to 0

For SPIs and PPIs, adds the pending state to interrupt number 32n + x. Reads and writes have the following behavior:

Set_pending_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not pending on any PE. If written, has no effect.
0b1	<ul style="list-style-type: none"> On this PE if the interrupt is an SGI or PPI. On at least one PE if the interrupt is an SPI. <p>If read, indicates that the corresponding interrupt is pending, or active and pending.</p> <p>If written, changes the state of the corresponding interrupt from inactive to pending, or from active to active and pending. This has no effect in the following cases:</p> <ul style="list-style-type: none"> If the interrupt is an SGI. The pending state of an SGI can be set using GICD_SPENDSGIR<n>. If the interrupt is not inactive and is not active. If the interrupt is already pending because of a write to GICD_ISPENDR<n>. If the interrupt is already pending because the corresponding interrupt signal is asserted. In this case, the interrupt remains pending if the interrupt signal is deasserted.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Accessing GICD_ISPENDR<n>

Set-pending bits for SGIs are read-only and ignore writes. The Set-pending bits for SGIs are provided as [GICD_SPENDSGIR<n>](#).

When affinity routing is enabled for the Security state of an interrupt:

- Bits corresponding to SGIs and PPIs are RAZ/WI, and equivalent functionality for SGIs and PPIs is provided by GICR_ISPENDR0.
- Bits corresponding to Group 0 and Group 1 Secure interrupts can only be set by Secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

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 bits that correspond to Group 0 or Secure Group 1 interrupts are accessible only by Secure accesses and are RAZ/WI to Non-secure accesses.

GICD_ISPENDR<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0200 + (4 * n)	GICD_ISPENDR<n>

- When [GICD_CTLR.DS](#)==0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses This on interface this is interface accessible areas follows: **RW**.

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

The GICD_ISPENDR<n>E characteristics are:

Purpose

Adds the pending state to the corresponding SPI in the extended SPI range.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICD_ISPENDR<n>E are RES0.

When `GICD_TYPER.ESPI=0`, these registers are RES0.

When [GICD_TYPER](#).ESPI==1, the number of implemented GICD_ISPENDR<n>E registers is ([GICD_TYPER](#).ESPI range+1). Registers are numbered from 0.

Attributes

GICD_ISPENDR<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25
Set_pending_bit31	Set_pending_bit30	Set_pending_bit29	Set_pending_bit28	Set_pending_bit27	Set_pending_bit26	Set_pending_bit25

Set_pending_bit<x>, bit [x], for x = 31 to 0

For the extended SPIs, adds the pending state to interrupt number x. Reads and writes have the following behavior:

Set_pending_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not pending. If written, has no effect.
0b1	If read, indicates that the corresponding interrupt is pending, or active and pending. If written, changes the state of the corresponding interrupt from inactive to pending, or from active to active and pending. This has no effect in the following cases: <ul style="list-style-type: none"> • If the interrupt is already pending because of a write to GICD_ISPENDR<n>E. • If the interrupt is already pending because the corresponding interrupt signal is asserted. In this case, the interrupt remains pending if the interrupt signal is deasserted.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m , when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_ISPENDR<n>E number, n, is given by $n = (m-4096) \text{ DIV } 32$.
- The offset of the required GICD_ISPENDR<n>E is $(0 \times 1600 + (4 * n))$.
- The bit number of the required group modifier bit in this register is $(m-4096) \text{ MOD } 32$.

Accessing GICD_ISPENDR<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICD_ISPENDR<n>E, the corresponding bit is RES0.

When `GICD_CTLR.DS==0`, bits corresponding to Secure SPIs are RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICD_ISPENDR<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x1600 + (4 * n)	GICD_ISPENDR<n>E

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: RW.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

GICD_ITARGETSR<n>, Interrupt Processor Targets Registers, n = 0 - 254

The GICD_ITARGETSR<n> characteristics are:

Purpose

When affinity routing is not enabled, holds the list of target PEs for the interrupt. That is, it holds the list of CPU interfaces to which the Distributor forwards the interrupt if it is asserted and has sufficient priority.

Configuration

These registers are available in all configurations of the GIC. When [GICD_CTLR.DS](#)==0, these registers are Common.

The number of implemented GICD_ITARGETSR<n> registers is 8*([GICD_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD_ITARGETSR0 to GICD_ITARGETSR7 are Banked for each connected PE with [GICR_TYPER.Processor_Number](#) < 8.

Accessing GICD_ITARGETSR0 to GICD_ITARGETSR7 from a PE with [GICR_TYPER.Processor_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

Attributes

GICD_ITARGETSR<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
CPU_targets_offset_3B								CPU_targets_offset_2B								CPU_targets_offset_1B								CPU_targets_offset_0B							

PEs in the system number from 0, and each bit in a PE targets field refers to the corresponding PE. For example, a value of 0x3 means that the Pending interrupt is sent to PEs 0 and 1. For GICD_ITARGETSR0-GICD_ITARGETSR7, a read of any targets field returns the number of the PE performing the read.

CPU_targets_offset_3B, bits [31:24]

PE targets for an interrupt, at byte offset 3.

The reset behavior of this field is:

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

CPU_targets_offset_2B, bits [23:16]

PE targets for an interrupt, at byte offset 2.

The reset behavior of this field is:

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

CPU_targets_offset_1B, bits [15:8]

PE targets for an interrupt, at byte offset 1.

The reset behavior of this field is:

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

CPU_targets_offset_0B, bits [7:0]

PE targets for an interrupt, at byte offset 0.

The reset behavior of this field is:

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

The bits that are set to 1 in the PE targets field determine which PEs are targeted:

Value of PE targets field	Interrupt targets
0bxxxxxx1	CPU interface 0
0bxxxxx1x	CPU interface 1
0bxxxx1xx	CPU interface 2
0bxxx1xxx	CPU interface 3
0bxx1xxxx	CPU interface 4
0bx1xxxxx	CPU interface 5
0b1xxxxxx	CPU interface 6
0b1xxxxxx	CPU interface 7

For interrupt ID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_ITARGETSR<n> number, n, is given by $n = m \text{ DIV } 4$.
- The offset of the required GICD_ITARGETSR<n> register is $(0x800 + (4 * n))$.
- The byte offset of the required Priority field in this register is $m \text{ MOD } 4$, where:
 - Byte offset 0 refers to register bits [7:0].
 - Byte offset 1 refers to register bits [15:8].
 - Byte offset 2 refers to register bits [23:16].
 - Byte offset 3 refers to register bits [31:24].

Software can write to these registers at any time. Any change to a targets field value:

- Has no effect on any active interrupt. This means that removing a CPU interface from a targets list does not cancel an active state for interrupts on that CPU interface. There is no effect on interrupts that are active and pending until the active status is cleared, at which time it is treated as a pending interrupt.
- Has an effect on any pending interrupts. This means:
 - Enables the CPU interface to be chosen as a target for the pending interrupt using an IMPLEMENTATION DEFINED mechanism.
 - Removing a CPU interface from the target list of a pending interrupt removes the pending state of the interrupt on that CPU interface.

Accessing GICD_ITARGETSR<n>

These registers are used when affinity routing is not enabled. When affinity routing is enabled for the Security state of an interrupt, the target PEs for an interrupt are defined by [GICD_IROUTER<n>](#) and the associated byte in GICD_ITARGETSR<n> is RES0. An implementation is permitted to make the byte RAZ/WI in this case.

- These registers are byte-accessible.
- A register field corresponding to an unimplemented interrupt is RAZ/WI.
- A field bit corresponding to an unimplemented CPU interface is RAZ/WI.
- GICD_ITARGETSR0-GICD_ITARGETSR7 are read-only. Each field returns a value that corresponds only to the PE reading the register.
- It is IMPLEMENTATION DEFINED which, if any, SPIs are statically configured in hardware. The field for such an SPI is read-only, and returns a value that indicates the PE targets for the 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 bits that correspond to Group 0 or Secure Group 1 interrupts are accessible only by Secure accesses and are RAZ/WI to Non-secure accesses.

In a single connected PE implementation, all interrupts target one PE, and these registers are RAZ/WI.

(old)

htmldiff from-

(new)

GICD_NSACR<n>, Non-secure Access Control Registers, n = 0 - 63

The GICD_NSACR<n> characteristics are:

Purpose

Enables Secure software to permit Non-secure software on a particular PE to create and control Group 0 interrupts.

Configuration

The concept of selective enabling of Non-secure access to Group 0 and Secure Group 1 interrupts applies to SGIs and SPIs.

GICD_NSACR0 is a Banked register used for SGIs. A copy is provided for every PE that has a CPU interface and that supports this feature.

Attributes

GICD_NSACR<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
NS_access15	NS_access14	NS_access13	NS_access12	NS_access11	NS_access10	NS_access9	NS_access8	NS_access7									

NS_access<x>, bits [2x+1:2x], for x = 15 to 0

Controls Non-secure access of the interrupt with ID 16n + x.

If the corresponding interrupt does not support configurable Non-secure access, the field is RAZ/WI.

Otherwise, the field is RW and determines the level of Non-secure control permitted if the interrupt is a Secure interrupt. If the interrupt is a Non-secure interrupt, this field is ignored.

The possible values of each 2-bit field are:

NS_access<x>	Meaning
0b00	No Non-secure access is permitted to fields associated with the corresponding interrupt.
0b01	Non-secure read and write access is permitted to set-pending bits in GICD_ISPENDR<n> associated with the corresponding interrupt. A Non-secure write access to GICD_SETSPI_NSR is permitted to set the pending state of the corresponding interrupt. A Non-secure write access to GICD_SGIR is permitted to generate a Secure SGI for the corresponding interrupt. An implementation might also provide read access to clear-pending bits in GICD_ICPENDR<n> associated with the corresponding interrupt.
0b10	As 0b01, but adds Non-secure read and write access permission to fields associated with the corresponding interrupt in the GICD_ICPENDR<n> registers. A Non-secure write access to GICD_CLRSPI_NSR is permitted to clear the pending state of the corresponding interrupt. Also adds Non-secure read access permission to fields associated with the corresponding interrupt in the GICD_ISACTIVER<n> and GICD_ICACTIVER<n> registers.
0b11	For GICD_NSACR0 this encoding is reserved and treated as 10. For all other GICD_NSACR<n> registers this encoding is treated as 0b10, but adds Non-secure read and write access permission to GICD_ITARGETSR<n> and GICD_IROUTER<n> fields associated with the corresponding interrupt.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For interrupt ID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_NSACR<n> number, n, is given by $n = m \text{ DIV } 16$.
- The offset of the required GICD_NSACR<n> register is $(0xE00 + (4*n))$.

Note

Because each field in this register comprises two bits, GICD_NSACR0 controls access rights to SGI registers, GICD_NSACR1 controls access to PPI registers (and is always RAZ/WI), and all other GICD_NSACR<n> registers control access to SPI registers.

For compatibility with GICv2, writes to GICD_NSACR0 for a particular PE must be coordinated within the Distributor and must update [GICR_NSACR](#) for the Redistributor associated with that PE.

Accessing GICD_NSACR<n>

These registers are always used when affinity routing is not enabled. When affinity routing is enabled for the Secure state, GICD_NSACR0 is RES0 and [GICR_NSACR](#) provides equivalent functionality for SGIs.

These registers do not support PPIs, therefore GICD_NSACR1 is RAZ/WI.

GICD_NSACR<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	$0xE00 + (4 * n)$	GICD_NSACR<n>

This interface is accessible as follows:

- When GICD_CTLR.DS == 1 accesses to this register are **RAZ/WI**.
- When GICD_CTLR.DS == 0 and an access is Secure accesses to this register are **RW**.
- When GICD_CTLR.DS == 0 and an access is Non-secure accesses to this register are **RAZ/WI**.
- When GICD_CTLR.DS == 0, FEAT_RME is implemented and IsAccessRoot() accesses to this register are **RW**.
- When GICD_CTLR.DS == 0, FEAT_RME is implemented and IsAccessRealm() accesses to this register are **RAZ/WI**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)htmldiff from-(new)

GICD_NSACR<n>E, Non-secure Access Control Registers, n = 0 - 63

The GICD_NSACR<n>E characteristics are:

Purpose

Enables Secure software to permit Non-secure software on a particular PE to create and control Group 0 interrupts.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICD_NSACR<n>E are RES0.

When [GICD_TYPER](#).ESPI==0, these registers are RES0.

When [GICD_TYPER](#).ESPI==1, the number of implemented GICD_ICFGR<n>E registers is (([GICD_TYPER](#).ESPI_range+1)*2). Registers are numbered from 0.

Attributes

GICD_NSACR<n>E 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
NS_access15	NS_access14	NS_access13	NS_access12	NS_access11	NS_access10	NS_access9	NS_access8	NS_access7									

NS_access<x>, bits [2x+1:2x], for x = 15 to 0

Controls Non-secure access of the interrupt with ID 16n + x.

If the corresponding interrupt does not support configurable Non-secure access, the field is RAZ/WI.

Otherwise, the field is RW and determines the level of Non-secure control permitted if the interrupt is a Secure interrupt. If the interrupt is a Non-secure interrupt, this field is ignored.

The possible values of each 2-bit field are:

NS_access<x>	Meaning
0b00	No Non-secure access is permitted to fields associated with the corresponding interrupt.
0b01	Non-secure read and write access is permitted to set-pending bits in GICD_ISPENDR<n>E associated with the corresponding interrupt. A Non-secure write access to GICD_SETSPL_NSR is permitted to set the pending state of the corresponding interrupt.
0b10	As 0b01, but adds Non-secure read and write access permission to fields associated with the corresponding interrupt in the GICD_ICPENDR<n>E registers. A Non-secure write access to GICD_CLRSPL_NSR is permitted to clear the pending state of the corresponding interrupt. Also adds Non-secure read access permission to fields associated with the corresponding interrupt in the GICD_ISACTIVER<n>E and GICD_ICACTIVER<n>E registers.
0b11	This encoding is treated as 0b10, but adds Non-secure read and write access permission to GICD_IROUTER<n>E fields associated with the corresponding interrupt.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For interrupt ID m , when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_NSACR<n>E number, n, is given by $n = (m - 4096) \text{ DIV } 16$.
- The offset of the required GICD_NSACR<n>E register is $(0x3600 + (4*n))$.

Accessing GLCD_NSACR<n>E

GICD_NSACR<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x3600 + (4 * n)	GICD_NSACR<n>E

This interface is accessible as follows:

- When `GICD_CTLR.DS == 1` accesses to this register are **RAZ/WI**.
- When `GICD_CTLR.DS == 0` and an access is Secure accesses to this register are **RW**.
- When `GICD_CTLR.DS == 0` and an access is Non-secure accesses to this register are **RAZ/WI**.
- When `GICD_CTLR.DS == 0`, `FEAT_RME` is implemented and `IsAccessRoot()` accesses to this register are **RW**.
- When `GICD_CTLR.DS == 0`, `FEAT_RME` is implemented and `IsAccessRealm()` accesses to this register are **RAZ/WI**.

3020/09/2021 14:53:37; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd7b36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICD_SETSPI_NSR, Set Non-secure SPI Pending Register

The GICD_SETSPI_NSR characteristics are:

Purpose

Adds the pending state to a valid SPI if permitted by the Security state of the access and the [GICD_NSACR<n>](#) value for that SPI.

A write to this register changes the state of an inactive SPI to pending, and the state of an active SPI to active and pending.

Configuration

If [GICD_TYPER](#).MBIS == 0, this register is reserved.

When [GICD_CTLR](#).DS == 1, this register provides functionality for all SPIs.

Attributes

GICD_SETSPI_NSR 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																INTID															

Bits [31:13]

Reserved, RES0.

INTID, bits [12:0]

The INTID of the SPI.

The function of this register depends on whether the targeted SPI is configured to be an edge-triggered or level-sensitive interrupt:

- For an edge-triggered interrupt, a write to GICD_SETSPI_NSR or [GICD_SETSPI_SR](#) adds the pending state to the targeted interrupt. It will stop being pending on activation, or if the pending state is removed by a write to [GICD_CLRSPI_NSR](#), [GICD_CLRSPI_SR](#), or [GICD_ICPENDR<n>](#).
- For a level-sensitive interrupt, a write to GICD_SETSPI_NSR or [GICD_SETSPI_SR](#) adds the pending state to the targeted interrupt. It will remain pending until it is deasserted by a write to [GICD_CLRSPI_NSR](#) or [GICD_CLRSPI_SR](#). If the interrupt is activated between having the pending state added and being deactivated, then the interrupt will be active and pending.

Accessing GICD_SETSPI_NSR

Writes to this register have no effect if:

- The value written specifies a Secure SPI, the value is written by a Non-secure access, and the value of the corresponding [GICD_NSACR<n>](#) register is 0.
- The value written specifies an invalid SPI.
- The SPI is already pending.

16-bit accesses to bits [15:0] of this register must be supported.

Note

A Secure access to this register can set the pending state of any valid SPI.

GICD_SETSPI_NSR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0040	GICD_SETSPI_NSR

- When GICD_CTLR.DS == 0 accesses to this register are **WO**.
- When an access is Secure accesses to this register are **WO**.
- When an access is Non-secure accesses to this register are **WO**.

Accesses This on interface this is interface accessible areas follows: **WO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd536e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICD_SETSPI_SR, Set Secure SPI Pending Register

The GICD_SETSPI_SR characteristics are:

Purpose

Adds the pending state to a valid SPI.

A write to this register changes the state of an inactive SPI to pending, and the state of an active SPI to active and pending.

Configuration

If [GICD_TYPER](#).MBIS == 0, this register is reserved.

When [GICD_CTLR](#).DS == 1, this register is WI.

Attributes

GICD_SETSPI_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
RES0																INTID															

Bits [31:13]

Reserved, RES0.

INTID, bits [12:0]

The INTID of the SPI.

The function of this register depends on whether the targeted SPI is configured to be an edge-triggered or level-sensitive interrupt:

- For an edge-triggered interrupt, a write to [GICD_SETSPI_NSR](#) or GICD_SETSPI_SR adds the pending state to the targeted interrupt. It will stop being pending on activation, or if the pending state is removed by a write to [GICD_CLRSPI_NSR](#), [GICD_CLRSPI_SR](#), or [GICD_ICPENDR<n>](#).
- For a level-sensitive interrupt, a write to [GICD_SETSPI_NSR](#) or GICD_SETSPI_SR adds the pending state to the targeted interrupt. It will remain pending until it is deasserted by a write to [GICD_CLRSPI_NSR](#) or [GICD_CLRSPI_SR](#). If the interrupt is activated between having the pending state added and being deactivated, then the interrupt will be active and pending.

Accessing GICD_SETSPI_SR

Writes to this register have no effect if:

- The value is written by a Non-secure access.
- The value written specifies an invalid SPI.
- The SPI is already pending.

16-bit accesses to bits [15:0] of this register must be supported.

GICD_SETSPI_SR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0050	GICD_SETSPI_SR

This interface is accessible as follows:

- When GICD_CTLR.DS == 10 accesses to this register are **WI**.
- When GICD_CTLR.DS == 0 and an access is Secure accesses to this register are **WO**.
- When GICD_CTLR.DS == 0 and an access is Non-secure accesses to this register are **WI**.
- When GICD_CTLR.DS == 0, FEAT_RME is implemented and IsAccessRoot() accesses to this register are **WO**.
- When GICD_CTLR.DS == 0, FEAT_RME is implemented and IsAccessRealm() accesses to this register are **WI**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICD_SGIR, Software Generated Interrupt Register

The GICD_SGIR characteristics are:

Purpose

Controls the generation of SGIs.

Configuration

This register is available in all configurations of the GIC. If the GIC supports two Security states this register is Common.

Attributes

GICD_SGIR 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						TargetListFilter						CPUTargetList						NSATT	RES0						INTID						

Bits [31:26]

Reserved, RES0.

TargetListFilter, bits [25:24]

Determines how the Distributor processes the requested SGI.

TargetListFilter	Meaning
0b00	Forward the interrupt to the CPU interfaces specified by GICD_SGIR.CPUTargetList.
0b01	Forward the interrupt to all CPU interfaces except that of the PE that requested the interrupt.
0b10	Forward the interrupt only to the CPU interface of the PE that requested the interrupt.
0b11	Reserved.

CPUTargetList, bits [23:16]

When GICD_SGIR.TargetListFilter is 0b00, this field defines the CPU interfaces to which the Distributor must forward the interrupt.

Each bit of the field refers to the corresponding CPU interface. For example, CPUTargetList[0] corresponds to interface 0. Setting a bit to 1 indicates that the interrupt must be forwarded to the corresponding interface.

If this field is 0b00000000 when GICD_SGIR.TargetListFilter is 0b00, the Distributor does not forward the interrupt to any CPU interface.

NSATT, bit [15]

Specifies the required group of the SGI.

NSATT	Meaning
0b0	Forward the SGI specified in the INTID field to a specified CPU interface only if the SGI is configured as Group 0 on that interface.
0b1	Forward the SGI specified in the INTID field to a specified CPU interface only if the SGI is configured as Group 1 on that interface.

This field is writable only by a Secure access. Non-secure accesses can also generate Group 0 interrupts, if allowed to do so by GICD_NSACR0. Otherwise, Non-secure writes to GICD_SGIR generate an SGI only if the specified SGI is programmed as Group 1, regardless of the value of bit [15] of the write.

Bits [14:4]

Reserved, RES0.

INTID, bits [3:0]

The INTID of the SGI to forward to the specified CPU interfaces.

Accessing GICD_SGIR

This register is used only when affinity routing is not enabled. When affinity routing is enabled, this register is RES0.

It is IMPLEMENTATION DEFINED whether this register has any effect when the forwarding of interrupts by the Distributor is disabled by [GICD_CTLR](#).

GICD_SGIR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0F00	GICD_SGIR

- When GICD_CTLR.DS == 0 accesses to this register are **WO**.
- When an access is Secure accesses to this register are **WO**.
- When an access is Non-secure accesses to this register are **WO**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **WO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

GICD_SPENDSGIR<n>, SGI Set-Pending Registers, n = 0 - 3

The GICD_SPENDSGIR<n> characteristics are:

Purpose

Adds the pending state to an SGI.

A write to this register changes the state of an inactive SGI to pending, and the state of an active SGI to active and pending.

Configuration

Four SGI set-pending registers are implemented. Each register contains eight set-pending bits for each of four SGIs, for a total of 16 possible SGIs.

In multiprocessor implementations, each PE has a copy of these registers.

Attributes

GICD_SPENDSGIR<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
SGI_set_pending_bits3								SGI_set_pending_bits2								SGI_set_pending_bits1								SGI_set_pending_bits0							

SGI_set_pending_bits<x>, bits [8x+7:8x], for x = 3 to 0

Adds the pending state to SGI number 4n + x for the PE corresponding to the bit number written to.

Reads and writes have the following behavior:

SGI_set_pending_bits<x>	Meaning
0x00	If read, indicates that the SGI from the corresponding PE is not pending and is not active and pending. If written, has no effect.
0x01	If read, indicates that the SGI from the corresponding PE is pending or is active and pending. If written, adds the pending state to the SGI for the corresponding PE.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For SGI ID m, generated by processing element C writing to the corresponding [GICD_SGIR](#) field, where DIV and MOD are the integer division and modulo operations:

- The corresponding GICD_SPENDSGIR<n> number is given by n = m DIV 4.
- The offset of the required register is (0xF20 + (4n)).
- The offset of the required field within the register GICD_SPENDSGIR<n> is given by m MOD 4.
- The required bit in the 8-bit SGI set-pending field m is bit C.

Accessing GICD_SPENDSGIR<n>

These registers are used only when affinity routing is not enabled. When affinity routing is enabled for the Security state of an interrupt then the bit associated with SGI in that Security state is RES0. An implementation is permitted to make the register RAZ/WI in this case.

A register bit that corresponds to an unimplemented SGI is RAZ/WI.

These registers are byte-accessible.

If the GIC implementation supports two Security states:

- A register bit that corresponds to a Group 0 interrupt is RAZ/WI to Non-secure accesses.
- Register bits corresponding to unimplemented PEs are RAZ/WI.

GICD_SPENDSGIR<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0F20 + (4 * n)	GICD_SPENDSGIR<n>

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RW**.

30/09/2021 14:53:37; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICD_STATUSR, Error Reporting Status Register

The GICD_STATUSR characteristics are:

Purpose

Provides software with a mechanism to detect:

- Accesses to reserved locations.
- Writes to read-only locations.
- Reads of write-only locations.

Configuration

If the GIC implementation supports two Security states this register is Banked to provide Secure and Non-secure copies.

Attributes

GICD_STATUSR 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																WROD				RWOD		WRD		RRD							

Bits [31:4]

Reserved, RES0.

WROD, bit [3]

Write to an RO location.

WROD	Meaning
0b0	Normal operation.
0b1	A write to an RO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

RWOD, bit [2]

Read of a WO location.

RWOD	Meaning
0b0	Normal operation.
0b1	A read of a WO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

WRD, bit [1]

Write to a reserved location.

WRD	Meaning
0b0	Normal operation.
0b1	A write to a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

RRD, bit [0]

Read of a reserved location.

RRD	Meaning
0b0	Normal operation.
0b1	A read of a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

Accessing GICD_STATUSR

This is an optional register. If the register is not implemented, the location is RAZ/WI.

GICD_STATUSR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0010	GICD_STATUSR (S)

This interface is accessible as follows:

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When FEAT_RME is implemented and IsAccessRoot() accesses to this register are **RW**.

Component	Frame	Offset	Instance
GIC Distributor	Dist_base	0x0010	GICD_STATUSR (NS)

This interface is accessible as follows:

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.
- When FEAT_RME is implemented and IsAccessRealm() accesses to this register are **RAZ/WI**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICD_TYPER, Interrupt Controller Type Register

The GICD_TYPER characteristics are:

Purpose

Provides information about what features the GIC implementation supports. It indicates:

- Whether the GIC implementation supports two Security states.
- The maximum number of INTIDs that the GIC implementation supports.
- The number of PEs that can be used as interrupt targets.

Configuration

This register is available in all configurations of the GIC. When [GICD_CTLR.DS](#)==0, this register is Common.

Attributes

GICD_TYPER 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
ESPI_range	RSS	No1N	A3V	IDbits	DVIS	LPIS	MBIS	num_LPis	SecurityExtn	NMI	RES0	ESPI	CPUNumber	ITLinesNumber																	

ESPI_range, bits [31:27]

When GICD_TYPER.ESPI == 1:

Indicates the maximum INTID in the Extended SPI range.

Maximum Extended SPI INTID is $(32 * (\text{ESPI_range} + 1) + 4095)$.

The ESPI_range field only indicates the maximum number of SPIs that the GIC implementation might support. This value determines the number of instances of the following interrupt registers:

- [GICD_IGROUPR<n>E](#).
- [GICD_ISENBALER<n>E](#).
- [GICD_ICENABLER<n>E](#).
- [GICD_ISPENDR<n>E](#).
- [GICD_ICPENDR<n>E](#).
- [GICD_ISACTIVER<n>E](#).
- [GICD_ICACTIVER<n>E](#).
- [GICD_IPRIORITYR<n>E](#).
- [GICD_ICFGR<n>E](#).
- [GICD_IROUTER<n>E](#).
- [GICD_IGRPMODR<n>E](#).

The GIC architecture does not require a GIC implementation to support a continuous range of SPI interrupt IDs. Software must check which SPI INTIDs are supported, up to the maximum value indicated by GICD_TYPER.ESPI_range.

Otherwise:

Reserved, RES0.

RSS, bit [26]

Range Selector Support.

RSS	Meaning
0b0	The IRI supports targeted SGIs with affinity level 0 values of 0 - 15.
0b1	The IRI supports targeted SGIs with affinity level 0 values of 0 - 255.

No1N, bit [25]

Indicates whether 1 of N SPI interrupts are supported.

No1N	Meaning
0b0	1 of N SPI interrupts are supported.
0b1	1 of N SPI interrupts are not supported.

A3V, bit [24]

Affinity 3 valid. Indicates whether the Distributor supports nonzero values of Affinity level 3.

A3V	Meaning
0b0	The Distributor only supports zero values of Affinity level 3.
0b1	The Distributor supports nonzero values of Affinity level 3.

IDbits, bits [23:19]

The number of interrupt identifier bits supported, minus one.

DVIS, bit [18]

When FEAT_GICv4 is implemented:

Indicates whether the implementation supports Direct Virtual LPI injection.

DVIS	Meaning
0b0	The implementation does not support Direct Virtual LPI injection.
0b1	The implementation supports Direct Virtual LPI injection.

Otherwise:

Reserved, RES0.

LPIS, bit [17]

Indicates whether the implementation supports LPIS.

LPIS	Meaning
0b0	The implementation does not support LPIS.
0b1	The implementation supports LPIS.

MBIS, bit [16]

Indicates whether the implementation supports message-based interrupts by writing to Distributor registers.

MBIS	Meaning
0b0	The implementation does not support message-based interrupts by writing to Distributor registers. The GICD_CLRSPI_NSR , GICD_SETSPI_NSR , GICD_CLRSPI_SR , and GICD_SETSPI_SR registers are reserved.
0b1	The implementation supports message-based interrupts by writing to the GICD_CLRSPI_NSR , GICD_SETSPI_NSR , GICD_CLRSPI_SR , or GICD_SETSPI_SR registers.

num_LPIs, bits [15:11]

Number of supported LPIs.

- 0b00000 Number of LPIs as indicated by GICD_TYPER.IDbits.
- All other values Number of LPIs supported is $2^{(\text{num_LPIs}+1)}$.
 - Available LPI INTIDs are $8192..(8192 + 2^{(\text{num_LPIs}+1)} - 1)$.
 - This field cannot indicate a maximum LPI INTID greater than that indicated by GICD_TYPER.IDbits.

When the supported INTID width is less than 14 bits, this field is RES0 and no LPIs are supported.

SecurityExtn, bit [10]

Indicates whether the GIC implementation supports two Security states:

When [GICD_CTLR](#).DS == 1, this field is RAZ.

SecurityExtn	Meaning
0b0	The GIC implementation supports only a single Security state.
0b1	The GIC implementation supports two Security states.

NMI, bit [9]

Non-maskable Interrupts.

~~Reserved, RES0.~~

NMI	Meaning
0b0	Non-maskable interrupt property not supported.
0b1	Non-maskable interrupt property is supported.

ESPI, bit [8]

Extended SPI.

ESPI	Meaning
0b0	Extended SPI range not implemented.
0b1	Extended SPI range implemented.

CPUNumber, bits [7:5]

Reports the number of PEs that can be used when affinity routing is not enabled, minus 1.

These PEs must be numbered contiguously from zero, but the relationship between this number and the affinity hierarchy from MPIDR is IMPLEMENTATION DEFINED. If the implementation does not support ARE being zero, this field is 000.

(old)

htmldiff from-

(new)

GICD_TYPER2, Interrupt Controller Type Register 2

The GICD_TYPER2 characteristics are:

Purpose

Provides information about which features the GIC implementation supports.

Configuration

This register is present only when FEAT_GICv4p1 is implemented. Otherwise, direct accesses to GICD_TYPER2 are RES0.

When [GICD_CTLR.DS](#) == 0, this register is Common.

Attributes

GICD_TYPER2 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																							nASSGICap	VIL	RES0	VID					

Bits [31:9]

Reserved, RES0.

nASSGICap, bit [8]

Indicates whether SGIs can be configured to not have an active state.

nASSGICap	Meaning
0b0	SGIs have an active state.
0b1	SGIs can be globally configured not to have an active state.

This bit is RES0 on implementations that support two Security states.

VIL, bit [7]

Indicates whether 16 bits of vPEID are implemented.

VIL	Meaning
0b0	GIC supports 16-bit vPEID.
0b1	GIC supports GICD_TYPER2.VID + 1 bits of vPEID.

Bits [6:5]

Reserved, RES0.

VID, bits [4:0]

When GICD_TYPER2.VIL == 1, the number of bits is equal to the bits of vPEID minus one.

GICD_TYPER2 can be accessed through the memory-mapped interfaces:

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

GICM_CLRSPI_NSR, Clear Non-secure SPI Pending Register

The GICM_CLRSPI_NSR characteristics are:

Purpose

Removes the pending state from a valid SPI if permitted by the Security state of the access and the [GICD_NSACR<n>](#) value for that SPI.

A write to this register changes the state of a pending SPI to inactive, and the state of an active and pending SPI to active.

Configuration

This register is present only when GICM_TYPER.CLR == 1. Otherwise, direct accesses to GICM_CLRSPI_NSR are RES0.

When [GICD_CTLR.DS](#) == 1, this register provides functionality for all SPIs.

Attributes

GICM_CLRSPI_NSR 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																				INTID											

Bits [31:13]

Reserved, RES0.

INTID, bits [12:0]

This field is an alias of [GICD_CLRSPI_NSR](#).

Accessing GICM_CLRSPI_NSR

Writes to this register have no effect if:

- The value written specifies a Secure SPI, the value is written by a Non-secure access, and the value of the corresponding [GICD_NSACR<n>](#) register is less than 0b10.
- The value written specifies an invalid SPI.
- The SPI is not pending.

16-bit accesses to bits [15:0] of this register must be supported.

Note

A Secure access to this register can clear the pending state of any valid SPI.

(old)

htmldiff from-

(new)

GICM_CLRSPI_SR, Clear Secure SPI Pending Register

The GICM_CLRSPI_SR characteristics are:

Purpose

Removes the pending state from a valid SPI.

A write to this register changes the state of a pending SPI to inactive, and the state of an active and pending SPI to active.

Configuration

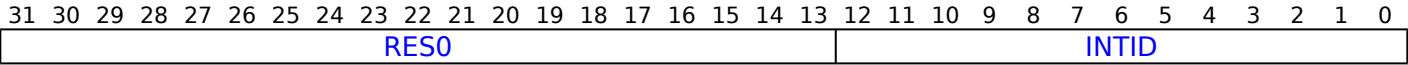
This register is present only when GICM_TYPER.SR == 1 and GICM_TYPER.CLR == 1. Otherwise, direct accesses to GICM_CLRSPI_SR are RES0.

When [GICD_CTLR.DS](#) == 1, this register is **WI**.

Attributes

GICM_CLRSPI_SR is a 32-bit register.

Field descriptions



Bits [31:13]

Reserved, RES0.

INTID, bits [12:0]

This field is an alias of [GICD_CLRSPI_SR](#).

Accessing GICM_CLRSPI_SR

Writes to this register have no effect if:

- The value is written by a Non-secure access.
- The value written specifies an invalid SPI.
- The SPI is not pending.

16-bit accesses to bits [15:0] of this register must be supported.

GICM_CLRSPI_SR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	MSI_base	0x0058	GICD_CLRSPI_SR

This interface is accessible as follows:

- When GICD_CTLR.DS == **10** accesses to this register are **WI**.

- When GICD_CTLR.DS == 0 and an access is Secure accesses to this register are **WO**.
- When GICD_CTLR.DS == 0 and an access is Non-secure accesses to this register are **WI**.
- When GICD_CTLR.DS == 0, FEAT_RME is implemented and IsAccessRoot() accesses to this register are **WO**.
- When GICD_CTLR.DS == 0, FEAT_RME is implemented and IsAccessRealm() accesses to this register are **WI**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICM_IIDR, Distributor Implementer Identification Register

The GICM_IIDR characteristics are:

Purpose

Provides information about the implementer and revision of the Distributor.

Configuration

This register is available in all configurations of the GIC. If the GIC implementation supports two Security states, this register is Common.

Attributes

GICM_IIDR 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
ProductID								RES0				Variant				Revision				Implementer											

ProductID, bits [31:24]

Product Identifier.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Bits [23:20]

Reserved, RES0.

Variant, bits [19:16]

Variant number. Typically, this field is used to distinguish product variants, or major revisions of a product.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Revision, bits [15:12]

Revision number. Typically, this field is used to distinguish minor revisions of a product.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Implementer, bits [11:0]

Contains the JEP106 code of the company that implemented the Distributor:

- Bits [11:8] are the JEP106 continuation code of the implementer. For an Arm implementation, this field is 0x4.
- Bit [7] is always 0.
- Bits [6:0] are the JEP106 identity code of the implementer. For an Arm implementation, bits [7:0] are therefore 0x3B.

Accessing GICM_IIDR

GICM_IIDR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	MSI_base	0x0FCC	GICM_IIDR

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

Accesses This on interface this is interface accessible are as follows: **RO**.

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICM_SETSPI_NSR, Set Non-secure SPI Pending Register

The GICM_SETSPI_NSR characteristics are:

Purpose

Adds the pending state to a valid SPI if permitted by the Security state of the access and the [GICD_NSACR<n>](#) value for that SPI.

A write to this register changes the state of an inactive SPI to pending, and the state of an active SPI to active and pending.

Configuration

When [GICD_CTLR.DS](#)==1, this register provides functionality for all SPIs.

Attributes

GICM_SETSPI_NSR 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																INTID															

Bits [31:13]

Reserved, RES0.

INTID, bits [12:0]

This field is an alias of [GICD_SETSPI_NSR](#).

Accessing GICM_SETSPI_NSR

Writes to this register have no effect if:

- The value written specifies a Secure SPI, the value is written by a Non-secure access, and the value of the corresponding [GICD_NSACR<n>](#) register is 0.
- The value written specifies an invalid SPI.
- The SPI is already pending.

16-bit accesses to bits [15:0] of this register must be supported.

Note

A Secure access to this register can set the pending state of any valid SPI.

GICM_SETSPI_NSR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
-----------	-------	--------	----------

GIC Distributor	MSI_base	0x0040	GICM_SETSPI_NSR
-----------------	----------	--------	-----------------

- When GICD_CTLR.DS == 0 accesses to this register are **WO**.
- When an access is Secure accesses to this register are **WO**.
- When an access is Non-secure accesses to this register are **WO**.

Accesses This on interface this is interface accessible are as follows: **WO**.

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

(old)

htmldiff from-

(new)

GICM_SETSPI_SR, Set Secure SPI Pending Register

The GICM_SETSPI_SR characteristics are:

Purpose

Adds the pending state to a valid SPI.

A write to this register changes the state of an inactive SPI to pending, and the state of an active SPI to active and pending.

Configuration

This register is present only when GICM_TYPER.SR == 1. Otherwise, direct accesses to GICM_SETSPI_SR are RES0.

When [GICD_CTLR.DS](#)==1, this register is WI.

Attributes

GICM_SETSPI_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
RES0																INTID															

Bits [31:13]

Reserved, RES0.

INTID, bits [12:0]

This field is an alias of [GICD_SETSPI_SR](#).

Accessing GICM_SETSPI_SR

Writes to this register have no effect if:

- The value is written by a Non-secure access.
- The value written specifies an invalid SPI.
- The SPI is already pending.

16-bit accesses to bits [15:0] of this register must be supported.

GICM_SETSPI_SR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	MSI_base	0x0050	GICM_SETSPI_SR

This interface is accessible as follows:

- When [GICD_CTLR.DS](#) == 10 accesses to this register are **WI**.
- When [GICD_CTLR.DS](#) == 0 and an access is Secure accesses to this register are **WO**.

- When GICD_CTLR.DS == 0 and an access is Non-secure accesses to this register are **WI**.
- When GICD_CTLR.DS == 0, FEAT_RME is implemented and IsAccessRoot() accesses to this register are **WO**.
- When GICD_CTLR.DS == 0, FEAT_RME is implemented and IsAccessRealm() accesses to this register are **WI**.

3020/09/2021 1412:5236; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICM_TYPER, Distributor MSI Type Register

The GICM_TYPER characteristics are:

Purpose

Provides information about what features the GIC implementation supports.

Configuration

This register is available in all configurations of the GIC. When [GICD_CTLR.DS](#)=0, this register is Common.

Attributes

GICM_TYPER 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
Valid	CLR	SR	INTID													RES0					NumSPIs										

Valid, bit [31]

Reports whether GICM_TYPER content is valid.

Valid	Meaning
0b0	GICM_TYPER reports no information on the capabilities of the GICM frame, all other fields are RES0.
0b1	GICM_TYPER reports information on capabilities of GICM frame.

CLR, bit [30]

Reports whether MSI clear registers are supported.

CLR	Meaning
0b0	MSI clear registers not implemented.
0b1	MSI clear registers implemented.

SR, bit [29]

Reports whether Secure aliases of MSI registers are supported.

SR	Meaning
0b0	Secure aliases of MSI registers not implemented.
0b1	Secure aliases of MSI registers implemented.

INTID, bits [28:16]

INTID of the first SPI assigned to this GICM frame.

Bits [15:11]

Reserved, RES0.

NumSPIs, bits [10:0]

Number of SPIs assigned to this GICM frame.

Accessing GICM_TYPER

GICM_TYPER can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Distributor	MSI_base	0x0004	GICM_TYPER

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

GICR_CLRLPIR, Clear LPI Pending Register

The GICR_CLRLPIR characteristics are:

Purpose

Clears the pending state of the specified LPI.

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_CLRLPIR is a 64-bit register.

Field descriptions

When GICR_TYPER.DirectLPI == 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															
																pINTID															
31	30	29	28	27	26	25	24	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.

pINTID, bits [31:0]

The INTID of the physical LPI.

Note

The size of this field is IMPLEMENTATION DEFINED, and is specified by the [GICD_TYPER.IDbits](#) field. Unimplemented bits are RES0.

When GICR_TYPER.DirectLPI == 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
																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.

(old)

htmldiff from-

(new)

GICR_CTLR, Redistributor Control Register

The GICR_CTLR characteristics are:

Purpose

Controls the operation of a Redistributor, and enables the signaling of LPis by the Redistributor to the connected PE.

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_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
UWP	RES0	DPG1S	DPG1NS	DPG0																											

UWP, bit [31]

Upstream Write Pending. Read-only. Indicates whether all upstream writes have been communicated to the Distributor.

UWP	Meaning
0b0	The effects of all upstream writes have been communicated to the Distributor, including any Generate SGI packets. For more information, see 'Generate SGI (ICC)' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).
0b1	Not all the effects of upstream writes, including any Generate SGI packets, have been communicated to the Distributor. For more information, see 'Generate SGI (ICC)' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Bits [30:27]

Reserved, RES0.

DPG1S, bit [26]

Disable Processor selection for Group 1 Secure interrupts. When [GICR_TYPER](#).DPGS == 1:

DPG1S	Meaning
0b0	A Group 1 Secure SPI configured to use the 1 of N distribution model can select this PE, if the PE is not asleep and if Secure Group 1 interrupts are enabled.
0b1	A Group 1 Secure SPI configured to use the 1 of N distribution model cannot select this PE.

When [GICR_TYPER](#).DPGS == 0 this bit is RAZ/WI.

When [GICD_CTLR.DS](#)==1, this field is RAZ/WI. In GIC implementations that support two Security states, this field is only accessible by Secure accesses, and is RAZ/WI to Non-secure accesses.

It is IMPLEMENTATION DEFINED whether these bits affect the selection of PEs for interrupts using the 1 of N distribution model when [GICD_CTLR.ARE_S](#)==0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

DPG1NS, bit [25]

Disable Processor selection for Group 1 Non-secure interrupts. When [GICR_TYPER.DPGS](#) == 1:

DPG1NS	Meaning
0b0	A Group 1 Non-secure SPI configured to use the 1 of N distribution model can select this PE, if the PE is not asleep and if Non-secure Group 1 interrupts are enabled.
0b1	A Group 1 Non-secure SPI configured to use the 1 of N distribution model cannot select this PE.

When [GICR_TYPER.DPGS](#) == 0 this bit is RAZ/WI.

It is IMPLEMENTATION DEFINED whether these bits affect the selection of PEs for interrupts using the 1 of N distribution model when [GICD_CTLR.ARE_NS](#)==0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

DPG0, bit [24]

Disable Processor selection for Group 0 interrupts. When [GICR_TYPER.DPGS](#) == 1:

DPG0	Meaning
0b0	A Group 0 SPI configured to use the 1 of N distribution model can select this PE, if the PE is not asleep and if Group 0 interrupts are enabled.
0b1	A Group 0 SPI configured to use the 1 of N distribution model cannot select this PE.

When [GICR_TYPER.DPGS](#) == 0 this bit is RAZ/WI.

When [GICD_CTLR.DS](#) == 1, this field is always accessible. In GIC implementations that support two Security states, this field is RAZ/WI to Non-secure accesses.

It is IMPLEMENTATION DEFINED whether these bits affect the selection of PEs for interrupts using the 1 of N distribution model when [GICD_CTLR.ARE_S](#) == 0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Bits [23:4]

Reserved, RES0.

RWP, bit [3]

Register Write Pending. This bit indicates whether a register write for the current Security state is in progress or not.

RWP	Meaning
0b0	The effect of all previous writes to the following registers are visible to all agents in the system: <ul style="list-style-type: none"> GICR_ICENABLER0 GICR_CTLR.DPG1S GICR_CTLR.DPG1NS GICR_CTLR.DPG0 GICR_CTLR, which clears EnableLPIs from 1 to 0. In FEAT_GICv4p1, GICR_VPROPBASER, which clears Valid from 1 to 0.
0b1	The effect of all previous writes to the following registers are not guaranteed by the architecture to be visible to all agents in the system while the changes are still being propagated: <ul style="list-style-type: none"> GICR_ICENABLER0 GICR_CTLR.DPG1S GICR_CTLR.DPG1NS GICR_CTLR.DPG0 GICR_CTLR, which clears EnableLPIs from 1 to 0. In FEAT_GICv4p1, GICR_VPROPBASER, which clears Valid from 1 to 0.

IR, bit [2]

LPI invalidate registers supported.

This bit is read-only.

IR	Meaning
0b0	This bit does not indicate whether the GICR_INVLPIR, GICR_INVALLR and GICR_SYNCRR are implemented or not.
0b1	GICR_INVLPIR, GICR_INVALLR and GICR_SYNCRR are implemented.

If [GICR_TYPER.DirectLPI](#) is 1 or [GICR_TYPER.RVPEI](#) is 1, [GICR_INVLPIR](#), [GICR_INVALLR](#), and [GICR_SYNCRR](#) are always implemented.

Arm recommends that implementations report GICR_CTLR.IR as 1 in these cases.

CES, bit [1]

Clear Enable Supported.

This bit is read-only.

CES	Meaning
0b0	The IRI does not indicate whether GICR_CTLR.EnableLPIs is RES1 once set.
0b1	GICR_CTLR.EnableLPIs is not RES1 once set.

Implementing GICR_CTLR.EnableLPIs as programmable and not reporting GICR_CTLR.CES == 1 is deprecated.

Implementing GICR_CTLR.EnableLPIs as RES1 once set is deprecated.

When GICR_CTLR.CES == 0, software cannot assume that GICR_CTLR.EnableLPIs is programmable without observing the bit being cleared.

EnableLPIs, bit [0]

In implementations where affinity routing is enabled for the Security state:

EnableLPIs	Meaning
0b0	LPI support is disabled. Any doorbell interrupt generated as a result of a write to a virtual LPI register must be discarded, and any ITS translation requests or commands involving LPIs in this Redistributor are ignored.
0b1	LPI support is enabled.

Note

If [GICR_TYPER](#).PLPIS == 0, this field is RES0. If [GICD_CTLR](#).ARE_NS is written from 1 to 0 when this bit is 1, behavior is an IMPLEMENTATION DEFINED choice between clearing GICR_CTLR.EnableLPis to 0 or maintaining its current value.

When affinity routing is not enabled for the Non-secure state, this bit is RES0.

When written from 0 to 1, the Redistributor loads the LPI Pending table from memory to check for any pending interrupts.

After it has been written to 1, it is IMPLEMENTATION DEFINED whether the bit becomes RES1 or can be cleared by to 0.

Where the bit remains programmable:

- Software must observe GICR_CTLR.RWP==0 after clearing GICR_CTLR.EnableLPis from 1 to 0 before writing [GICR_PENDBASER](#) or [GICR_PROPBASER](#), otherwise behavior is UNPREDICTABLE.
- Software must observe GICR_CTLR.RWP==0 after clearing GICR_CTLR.EnableLPis from 1 to 0 before setting GICR_CTLR.EnableLPis to 1, otherwise behavior is UNPREDICTABLE.

Note

If one or more ITS is implemented, Arm strongly recommends that all LPis are mapped to another Redistributor before GICR_CTLR.EnableLPis is cleared to 0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

The participation of a PE in the 1 of N distribution model for a given interrupt group is governed by the concatenation of [GICR_WAKER](#).ProcessorSleep, the appropriate [GICR_CTLR](#).DPG{1, 0} bit, and the PE interrupt group enable. The behavior options are:

PS	DPG{1S, 1NS, 0}	Enable	PE Behavior
0b0	0b0	0b0	The PE cannot be selected.
0b0	0b0	0b1	The PE can be selected.
0b0	0b1	*	The PE cannot be selected.
0b1	*	*	The PE cannot be selected when GICD_CTLR .E1NWF == 0. When GICD_CTLR .E1NWF == 1, the mechanism by which PEs are selected is IMPLEMENTATION DEFINED.

If an SPI using the 1 of N distribution model has been forwarded to the PE, and a write to GICR_CTLR occurs that changes the DPG bit for the interrupt group of the SPI, the IRI must attempt to select a different target PE for the SPI. This might have no effect on the forwarded SPI if it has already been activated.

Accessing GICR_CTLR

GICR_CTLR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	RD_base	0x0000	GICR_CTLR

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses on this interface are as follows: **RW**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

GIC Redistributor	SGI_base	0x0380	GICR_ICACTIVER0
----------------------	----------	--------	-----------------

- ~~When GICD_CTLR.DS == 0 accesses to this register are **RW**.~~
- ~~When an access is Secure accesses to this register are **RW**.~~
- ~~When an access is Non-secure accesses to this register are **RW**.~~

~~Accesses~~This oninterface ~~thisis~~ interfaceaccessible areas follows: **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd5b36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)htmldiff from-(new)

GICR_ICACTIVER<n>E, Interrupt Clear-Active Registers, n = 1 - 2

The GICR_ICACTIVER<n>E characteristics are:

Purpose

Removes the active state from the corresponding PPI.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICR_ICACTIVER<n>E are RES0.

A copy of this register is provided for each Redistributor.

Attributes

GICR_ICACTIVER<n>E 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
Clear_active_bit31	Clear_active_bit30	Clear_active_bit29	Clear_active_bit28	Clear_active_bit27	Clear_active_bit26	Clear_active_bit25	Clear_active_bit24	Clear_active_bit23	Clear_active_bit22	Clear_active_bit21	Clear_active_bit20	Clear_active_bit19	Clear_active_bit18	Clear_active_bit17	Clear_active_bit16	Clear_active_bit15	Clear_active_bit14	Clear_active_bit13	Clear_active_bit12	Clear_active_bit11	Clear_active_bit10	Clear_active_bit9	Clear_active_bit8	Clear_active_bit7	Clear_active_bit6	Clear_active_bit5	Clear_active_bit4	Clear_active_bit3	Clear_active_bit2	Clear_active_bit1	Clear_active_bit0

Clear_active_bit<x>, bit [x], for x = 31 to 0

For the extended PPIs, removes the active state to interrupt number x. Reads and writes have the following behavior:

Clear_active_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not active, and is not active and pending. If written, has no effect.
0b1	If read, indicates that the corresponding interrupt is active, or is active and pending. If written, deactivates the corresponding interrupt, if the interrupt is active. If the interrupt is already deactivated, the write has no effect.

The reset behavior of this field is:

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

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICR_ICACTIVER<n>E number, n, is given by $n = (m-1024) \text{ DIV } 32$.
- The offset of the required GICR_ICACTIVER<n>E is $(0x200 + (4*n))$.
- The bit number of the required group modifier bit in this register is $(m-1024) \text{ MOD } 32$.

Accessing GICR_ICACTIVER<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICR_ICACTIVER<n>E, the corresponding bit is RES0.

The GICR_ICENABLER0 characteristics are:

Purpose

Disables forwarding of the corresponding SGI or PPI to the CPU interfaces.

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR ICENABLER0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26
Clear_enable bit31	Clear_enable bit30	Clear_enable bit29	Clear_enable bit28	Clear_enable bit27	Clear_enable bit26

Clear_enable_bit<x>, bit [x], for x = 31 to 0

For PPIs and SGIs, controls the forwarding of interrupt number `x` to the CPU interfaces. Reads and writes have the following behavior:

Clear_enable_bit<x>	Meaning
0b0	If read, indicates that forwarding of the corresponding interrupt is disabled. If written, has no effect.
0b1	If read, indicates that forwarding of the corresponding interrupt is enabled. If written, disables forwarding of the corresponding interrupt. After a write of 1 to this bit, a subsequent read of this bit returns 0.

The reset behavior of this field is:

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

Accessing GICR_ICENABLER0

When affinity routing is not enabled for the Security state of an interrupt in GICR_ICENABLER0, the corresponding bit is RAZ/WI and equivalent functionality is provided by [GICD_ICENABLER<n>](#) with n=0.

This register only applies to SGIs (bits [15:0]) and PPIs (bits [31:16]). For SPIs, this functionality is provided by [GICD_ICENABLER<n>](#).

When `GICD_CTLR.DS == 0`, bits corresponding to Secure SGIs and PPIs are RAZ/WI to Non-secure accesses.

GICR_ICENABLER0 can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
-----------	-------	--------	----------

GIC Redistributor	SGI_base	0x0180	GICR_ICENABLER0
----------------------	----------	--------	-----------------

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses This on interface this interface accessible are as follows: **RW**.

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICR_ICENABLER<n>E, Interrupt Clear-Enable Registers, n = 1 - 2

The GICR_ICENABLER<n>E characteristics are:

Purpose

Disables forwarding of the corresponding PPI to the CPU interfaces.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICR_ICENABLER<n>E are RES0.

A copy of this register is provided for each Redistributor.

Attributes

GICR_ICENABLER<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	26
Clear_enable_bit31	Clear_enable_bit30	Clear_enable_bit29	Clear_enable_bit28	Clear_enable_bit27	Clear_enable_bit26

Clear_enable_bit<x>, bit [x], for x = 31 to 0

For the extended PPI range, controls the forwarding of interrupt number x to the CPU interface. Reads and writes have the following behavior:

Clear_enable_bit<x>	Meaning
0b0	If read, indicates that forwarding of the corresponding interrupt is disabled. If written, has no effect.
0b1	If read, indicates that forwarding of the corresponding interrupt is enabled. If written, disables forwarding of the corresponding interrupt. After a write of 1 to this bit, a subsequent read of this bit returns 0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICR_ICENABLER<n>E number, n, is given by $n = (m-1024) \text{ DIV } 32$.
- The offset of the required GICR_ICENABLER<n>E is $(0 \times 180 + (4 \times n))$.
- The bit number of the required group modifier bit in this register is $(m-1024) \text{ MOD } 32$.

Accessing GICR_ICENABLER<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICR_ICENABLER<n>E, the corresponding bit is RES0.

(old)

htmldiff from-

(new)

GICR_ICFGR0, Interrupt Configuration Register 0

The GICR_ICFGR0 characteristics are:

Purpose

Determines whether the corresponding SGI is edge-triggered or level-sensitive.

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_ICFGR0 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
Int_config15	Int_config14	Int_config13	Int_config12	Int_config11	Int_config10	Int_config9	Int_config8	Int_config7	Int_config6	Int_config5	Int_config4	Int_config3	Int_config2	Int_config1	Int_config0	Int_config0	Int_config0	Int_config0

Int_config<x>, bits [2x+1:2x], for x = 15 to 0

Indicates whether the is level-sensitive or edge-triggered.

Int_config<x>	Meaning
0b00	Corresponding interrupt is level-sensitive.
0b10	Corresponding interrupt is edge-triggered.

For SGIs, arethis field always indicates edge-triggered.

When the interrupt is visible to the current Security state, a read of this bit always returns the correct value to indicate the interrupt triggering method.

The reset behavior of this field is:

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

Accessing GICR_ICFGR0

This register is used when affinity routing is enabled.

When affinity routing is disabled for the Security state of an interrupt, the field for that interrupt is RES0 and an implementation is permitted to make the field RAZ/WI in this case. Equivalent functionality is provided by GICD_ICFGR<n> with n=0.

When GICD_CTLR.DS==0, a register bit that corresponds to a Group 0 or Secure Group 1 interrupt is RAZ/WI to Non-secure accesses.

GICR_ICFGR0 can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	SGI_base	0x0C00	GICR_ICFGR0

- When GICD_CTLR.DS == 0 accesses to this register are RW.

- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses This on interface this is interface accessible areas follows: **RW**.

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICR_ICFGR1, Interrupt Configuration Register 1

The GICR_ICFGR1 characteristics are:

Purpose

Determines whether the corresponding PPI is edge-triggered or level-sensitive.

Configuration

A copy of this register is provided for each Redistributor.

For each supported PPI, it is IMPLEMENTATION DEFINED whether software can program the corresponding Int_config field.

Changing Int_config when the interrupt is individually enabled is UNPREDICTABLE.

Changing the interrupt configuration between level-sensitive and edge-triggered (in either direction) at a time when there is a pending interrupt will leave the interrupt in an UNKNOWN pending state.

Attributes

GICR_ICFGR1 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
Int_config15	Int_config14	Int_config13	Int_config12	Int_config11	Int_config10	Int_config9	Int_config8	Int_config7	Int_config6	Int_config5	Int_config4	Int_config3	Int_config2	Int_config1	Int_config0	RES0	RES0	RES0

Int_config<x>, bits [2x+1:2x], for x = 15 to 0

Indicates whether the interrupt is level-sensitive or edge-triggered.

Int_config<x>	Meaning
0b00	Corresponding interrupt is level-sensitive.
0b10	Corresponding interrupt is edge-triggered.

Int_config[0] (bit [2x]) is RES0.

A read of this bit always returns the correct value to indicate the interrupt triggering method.

The reset behavior of this field is:

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

Accessing GICR_ICFGR1

This register is used when affinity routing is enabled.

When affinity routing is disabled for the Security state of an interrupt, the field for that interrupt is RES0 and an implementation is permitted to make the field RAZ/WI in this case. Equivalent functionality is provided by GICD_ICFGR<n> with n=1 .

When GICD_CTLR.DS==0, a register bit that corresponds to a Group 0 or Secure Group 1 interrupt is RAZ/WI to Non-secure accesses.

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

30/09/2021 14:52:37; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

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

(old)

htmldiff from-

(new)

GICR_ICFGR<n>E, Interrupt configuration registers, n = 2 - 5

The GICR_ICFGR<n>E characteristics are:

Purpose

Determines whether the corresponding PPI in the extended PPI range is edge-triggered or level-sensitive.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICR_ICFGR<n>E are RES0.

A copy of this register is provided for each Redistributor.

Attributes

GICR_ICFGR<n>E 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
Int_config15	Int_config14	Int_config13	Int_config12	Int_config11	Int_config10	Int_config9	Int_config8	Int_config7	Int_config6	Int_config5	Int_config4	Int_config3	Int_config2	Int_config1	Int_config0	Int_config0	Int_config0	Int_config0

Int_config<x>, bits [2x+1:2x], for x = 15 to 0

Indicates whether the interrupt is level-sensitive or edge-triggered.

Int_config[0] (bit [2x]) is RES0.

Int_config<x>	Meaning
0b00	The corresponding interrupt is level-sensitive.
0b10	The corresponding interrupt is edge-triggered.

The reset behavior of this field is:

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

For each supported extended PPI, it is IMPLEMENTATION DEFINED whether software can program the corresponding Int_config field.

Accessing GICR_ICFGR<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICR_ICFGR<n>E, the corresponding bit is RES0.

When [GICD_CTLR.DS](#)=0, a register bit that corresponds to a Group 0 or Secure Group 1 interrupt is RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICR_ICFGR<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
-----------	-------	--------	----------

GIC Redistributor	SGI_base	0x0C00 + (4 * n)	GICR_ICFGR<n>E
----------------------	----------	---------------------	----------------

- ~~When GICD_CTLR.DS == 0 accesses to this register are **RW**.~~
- ~~When an access is Secure accesses to this register are **RW**.~~
- ~~When an access is Non-secure accesses to this register are **RW**.~~

AccessesThis oninterface thisis interfaceaccessible areas follows: **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd5b36e47856e443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)htmldiff from-(new)

GICR_ICPENDR0, Interrupt Clear-Pending Register 0

The GICR_ICPENDR0 characteristics are:

Purpose

Removes the pending state from the corresponding SGI or PPI.

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_ICPENDR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26
Clear_pending_bit31	Clear_pending_bit30	Clear_pending_bit29	Clear_pending_bit28	Clear_pending_bit27	Clear_pending_bit26

Clear_pending_bit<x>, bit [x], for x = 31 to 0

Removes the pending state from interrupt number x. Reads and writes have the following behavior:

Clear_pending_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not pending. If written, has no effect.
0b1	If read, indicates that the corresponding interrupt is pending, or active and pending. If written, changes the state of the corresponding interrupt from pending to inactive, or from active and pending to active. This has no effect in the following cases: <ul style="list-style-type: none">If the interrupt is not pending and is not active and pending.If the interrupt is a level-sensitive interrupt that is pending or active and pending for a reason other than a write to GICD_ISPENDR<n>. In this case, if the interrupt signal continues to be asserted, the interrupt remains pending or active and pending.

The reset behavior of this field is:

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

Accessing GICR_ICPENDR0

When affinity routing is not enabled for the Security state of an interrupt in GICR_ICPENDR0, the corresponding bit is RAZ/WI and equivalent functionality is provided by [GICD_ICPENDR<n>](#) with n=0.

This register only applies to SGIs (bits [15:0]) and PPIs (bits [31:16]). For SPIs, this functionality is provided by [GICD_ICENABLER<n>](#).

When [GICD_CTLR.DS](#) == 0, bits corresponding to Secure SGIs and PPIs are RAZ/WI to Non-secure accesses.

GICR_ICPENDR0 can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	SGI_base	0x0280	GICR_ICPENDR0

- ~~When GICD_CTLR.DS == 0 accesses to this register are **RW**.~~
- ~~When an access is Secure accesses to this register are **RW**.~~
- ~~When an access is Non-secure accesses to this register are **RW**.~~

~~Accesses~~~~This on~~~~interface~~~~this is~~~~interface~~~~accessible~~ ~~areas follows:~~ **RW**.

~~3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea~~

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)htmldiff from-(new)

(old)htmldiff from-(new)

GICR_ICPENDR<n>E, Interrupt Clear-Pending Registers, n = 1 - 2

The GICR_ICPENDR<n>E characteristics are:

Purpose

Removes the pending state from the corresponding PPI.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICR_ICPENDR<n>E are RES0.

A copy of this register is provided for each Redistributor.

Attributes

GICR_ICPENDR<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	26
Clear_pending_bit31	Clear_pending_bit30	Clear_pending_bit29	Clear_pending_bit28	Clear_pending_bit27	Clear_pending_bit26

Clear_pending_bit<x>, bit [x], for x = 31 to 0

For the extended PPIs, removes the pending state to interrupt number x. Reads and writes have the following behavior:

Clear_pending_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not pending on this PE. If written, has no effect.
0b1	If read, indicates that the corresponding interrupt is pending, or active and pending on this PE. If written, changes the state of the corresponding interrupt from pending to inactive, or from active and pending to active. This has no effect in the following cases: <ul style="list-style-type: none">• If the interrupt is not pending and is not active and pending.• If the interrupt is a level-sensitive interrupt that is pending or active and pending for a reason other than a write to GICR_ICPENDR<n>E. In this case, if the interrupt signal continues to be asserted, the interrupt remains pending or active and pending.

The reset behavior of this field is:

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

GICR_IGROUPR0, Interrupt Group Register 0

The GICR_IGROUPR0 characteristics are:

Purpose

Controls whether the corresponding SGI or PPI is in Group 0 or Group 1.

Configuration

This register is available in all GIC configurations. If the GIC implementation supports two Security states, this register is Secure.

A copy of this register is provided for each Redistributor.

Attributes

GICR_IGROUPR0 is a 32-bit register.

Field descriptions

31	30	29	
Redistributor_group_status_bit31	Redistributor_group_status_bit30	Redistributor_group_status_bit29	Redistributor_g

Redistributor_group_status_bit<x>, bit [x], for x = 31 to 0

Group status bit. In this register:

- Bits [31:16] are group status bits for PPIs.
- Bits [15:0] are group status bits for SGIs.

Redistributor_group_status_bit<x>	Meaning
0b0	When GICD_CTLR.DS ==1, the corresponding interrupt is Group 0. When GICD_CTLR.DS ==0, the corresponding interrupt is Secure.
0b1	When GICD_CTLR.DS ==1, the corresponding interrupt is Group 1. When GICD_CTLR.DS ==0, the corresponding interrupt is Non-secure Group 1.

When [GICD_CTLR.DS](#) == 0, the bit that corresponds to the interrupt is concatenated with the equivalent bit in [GICR_IGRPMODR0](#) to form a 2-bit field that defines an interrupt group. The encoding of this field is at [GICR_IGRPMODR0](#).

The reset behavior of this field is:

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

The considerations for the reset value of this register are the same as those for [GICD_IGROUPR<n>](#) with n=0.

Page 1452

(old)

htmldiff from-

(new)

GICR_IGROUPR<n>E, Interrupt Group Registers, n = 1 - 2

The GICR_IGROUPR<n>E characteristics are:

Purpose

Controls whether the corresponding PPI is in Group 0 or Group 1.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICR_IGROUPR<n>E are RES0.

When [GICD_CTLR.DS](#)==0, this register is Secure.

A copy of this register is provided for each Redistributor.

Attributes

GICR_IGROUPR<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	26
Group_status_bit31	Group_status_bit30	Group_status_bit29	Group_status_bit28	Group_status_bit27	Group_status_bit26

Group_status_bit<x>, bit [x], for x = 31 to 0

Group status bit.

Group_status_bit<x>	Meaning
0b0	When GICD_CTLR.DS ==1, the corresponding interrupt is Group 0. When GICD_CTLR.DS ==0, the corresponding interrupt is Secure.
0b1	When GICD_CTLR.DS ==1, the corresponding interrupt is Group 1. When GICD_CTLR.DS ==0, the corresponding interrupt is Non-secure Group 1.

The reset behavior of this field is:

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

If affinity routing is enabled for the Security state of an interrupt, the bit that corresponds to the interrupt is concatenated with the equivalent bit in GICR_IGRPMODR<n>E to form a 2-bit field that defines an interrupt group. The encoding of this field is described in GICR_IGRPMODR<n>E.

If affinity routing is disabled for the Security state of an interrupt, the bit is RES0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICR_IGROUPR<n>E number, n, is given by $n = (m-1024) \text{ DIV } 32$.
- The offset of the required GICR_IGROUPR<n>E is $(0x080 + (4*n))$.
- The bit number of the required group modifier bit in this register is $(m-1024) \text{ MOD } 32$.

(old)

htmldiff from-

(new)

GICR_IGRPMODR0, Interrupt Group Modifier Register 0

The GICR_IGRPMODR0 characteristics are:

Purpose

When [GICD_CTLR.DS](#)==0, this register together with the [GICR_IGROUPR0](#) register, controls whether the corresponding interrupt is in:

- Secure Group 0.
- Non-secure Group 1.
- When System register access is enabled, Secure Group 1.

Configuration

When [GICD_CTLR.DS](#)==0, this register is Secure.

A copy of this register is provided for each Redistributor.

Attributes

GICR_IGRPMODR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	
Group_modifier_bit31	Group_modifier_bit30	Group_modifier_bit29	Group_modifier_bit28	Group_modifier_bit27	Group_modifier_bit26

Group_modifier_bit<x>, bit [x], for x = 31 to 0

Group modifier bit. In implementations where affinity routing is enabled for the Security state of an interrupt, the bit that corresponds to the interrupt is concatenated with the equivalent bit in [GICR_IGROUPR0](#) to form a 2-bit field that defines an interrupt group:

Group modifier bit	Group status bit	Definition	Short name
0b0	0b0	Secure Group 0	G0S
0b0	0b1	Non-secure Group 1	G1NS
0b1	0b0	Secure Group 1	G1S
0b1	0b1	Reserved, treated as Non-secure Group 1	-

The reset behavior of this field is:

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

Accessing GICR_IGRPMODR0

When affinity routing is not enabled for the Security state of an interrupt in GICR_IGRPMODR0, the corresponding bit is RES0 and equivalent functionality is provided by [GICD_IGRPMODR<n>](#) with n=0.

This register only applies to SGIs (bits [15:0]) and PPIs (bits [31:16]). For SPIs, this functionality is provided by [GICD_IGRPMODR<n>](#).

When [GICD_CTLR.ARE_S](#) == 0 or [GICD_CTLR.DS](#) == 1, GICR_IGRPMODR0 is RES0. An implementation can make this register RAZ/WI in this case.

When `GICD_CTLR.DS==0`, the register is RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

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 interrupt is neither lost nor handled more than one time. The effect of the change must be visible in finite time.

GICR_IGRPMODR0 can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	SGI_base	0x0D00	GICR_IGRPMODR0

- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses This on interface this is interface accessible areas follows: **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICR_IGRPMODR<n>E, Interrupt Group Modifier Registers, n = 1 - 2

The GICR_IGRPMODR<n>E characteristics are:

Purpose

When [GICD_CTLR.DS](#)==0, this register together with the GICR_IGROUPR<n>E registers, controls whether the corresponding interrupt is in:

- Secure Group 0.
- Non-secure Group 1.
- When System register access is enabled, Secure Group 1.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICR_IGRPMODR<n>E are RES0.

When [GICD_CTLR.DS](#)==0, this register is Secure.

A copy of this register is provided for each Redistributor.

Attributes

GICR_IGRPMODR<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	
Group_modifier_bit31	Group_modifier_bit30	Group_modifier_bit29	Group_modifier_bit28	Group_modifier_bit27	Group_modifier_bit26

Group_modifier_bit<x>, bit [x], for x = 31 to 0

Group modifier bit. In implementations where affinity routing is enabled for the Security state of an interrupt, the bit that corresponds to the interrupt is concatenated with the equivalent bit in [GICR_IGROUPR<n>E](#) to form a 2-bit field that defines an interrupt group:

Group modifier bit	Group status bit	Definition	Short name
0b0	0b0	Secure Group 0	G0S
0b0	0b1	Non-secure Group 1	G1NS
0b1	0b0	Secure Group 1	G1S
0b1	0b1	Reserved, treated as Non-secure Group 1	-

The reset behavior of this field is:

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

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICR_IGRPMODR<n>E number, n, is given by $n = (m-1024) \text{ DIV } 32$.
- The offset of the required GICR_IGRPMODR<n>E is $(0xD00 + (4*n))$.
- The bit number of the required group modifier bit in this register is $(m-1024) \text{ MOD } 32$.

Accessing GICR_IGRPMODR<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICR_IGRPMODR<n>E, the corresponding bit is RES0.

When `GICD_CTLR.DS==0`, the register is RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICR_IGRPMODR<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	SGI_base	0x0D00 + (4 * n)	GICR_IGRPMODR<n>E

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: RW.

30/09/2021 14:52:37; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICR_IIDR, Redistributor Implementer Identification Register

The GICR_IIDR characteristics are:

Purpose

Provides information about the implementer and revision of the Redistributor.

Configuration

This register is available in all configurations of the GIC. If the GIC implementation supports two Security states, this register is Common.

Attributes

GICR_IIDR 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
ProductID								RES0				Variant				Revision				Implementer											

ProductID, bits [31:24]

Product Identifier.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Bits [23:20]

Reserved, RES0.

Variant, bits [19:16]

Variant number. Typically, this field is used to distinguish product variants, or major revisions of a product.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Revision, bits [15:12]

Revision number. Typically, this field is used to distinguish minor revisions of a product.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Implementer, bits [11:0]

Contains the JEP106 code of the company that implemented the Redistributor:

- Bits [11:8] are the JEP106 continuation code of the implementer. For an Arm implementation, this field is 0x4.
- Bit [7] is always 0.
- Bits [6:0] are the JEP106 identity code of the implementer. For an Arm implementation, bits [7:0] are therefore 0x3B.

Accessing GICR_IIDR

GICR_IIDR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	RD_base	0x0004	GICR_IIDR

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

Accesses This on interface this is interface accessible are as follows: **RO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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.

Accessing GICR_INMIR0

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICR_INMIR0 can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	SGI_base	0x0F80	GICR_INMIR0

Accesses on this interface are **RW**.

no old file

htmldiff from-

(new)

no old file

htmldiff from-

(new)

GICR_INMIR<n>E, Non-maskable Interrupt Registers for Extended PPIs, x = 1 to 2., n = 1 - 2

The GICR_INMIR<n>E characteristics are:

Purpose

Controls whether the corresponding Extended PPI has the non-maskable property.

Configuration

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.

Attributes

GICR_INMIR<n>E 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.

Accessing GICR_INMIR<n>E

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICR_INMIR<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	SGI_base	0x0F80 + (4 * n)	GICR_INMIR<n>E

Accesses on this interface are **RW**.

no old file

htmldiff from-

(new)

GICR_INVALLR, Redistributor Invalidate All Register

The GICR_INVALLR characteristics are:

Purpose

Invalidates any cached configuration data of all physical LPIs, causing the GIC to reload the interrupt configuration from the physical LPI Configuration table at the address specified by [GICR_PROPBASER](#).

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_INVALLR is 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
V	RES0															vPEID															
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

V, bit [63]
When FEAT_GICv4p1 is implemented:

Indicates whether the INTID is virtual or physical.

V	Meaning
0b0	Invalidate is for a physical INTID.
0b1	Invalidate is for a virtual INTID.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

vPEID, bits [47:32]
When FEAT_GICv4p1 is implemented:

When GICR_INVLPIR.V == 0, this field is RES0

When GICR_INVLPIR.V == 1, this field is the target vPEID of the invalidate.

Note

The size of this field is IMPLEMENTATION DEFINED, and is specified by the [GICD_TYPER2.VIL](#) and [GICD_TYPER2.VID](#) fields. Unimplemented bits are RES0.

(old)

htmldiff from-

(new)

GICR_INVLPIR, Redistributor Invalidate LPI Register

The GICR_INVLPIR characteristics are:

Purpose

Invalidates the cached configuration data of a specified LPI, causing the GIC to reload the interrupt configuration from the physical LPI Configuration table at the address specified by [GICR_PROPBASER](#).

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_INVLPIR is 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
V	RES0															vPEID															
INTID																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

V, bit [63]
When FEAT_GICv4p1 is implemented:

Indicates whether the INTID is virtual or physical.

V	Meaning
0b0	Invalidate is for a physical INTID.
0b1	Invalidate is for a virtual INTID.

Otherwise:

Reserved, RES0.

Bits [62:48]

Reserved, RES0.

vPEID, bits [47:32]
When FEAT_GICv4p1 is implemented:

When GICR_INVLPIR.V == 0, this field is RES0
When GICR_INVLPIR.V == 1, this field is the target vPEID of the invalidate.

Note

The size of this field is IMPLEMENTATION DEFINED, and is specified by the [GICD_TYPER2.VIL](#) and [GICD_TYPER2.VID](#) fields. Unimplemented bits are RES0.

GICR_IPRIORITYR<n>, Interrupt Priority Registers, n = 0 - 7

The GICR_IPRIORITYR<n> characteristics are:

Purpose

Holds the priority of the corresponding interrupt for each SGI and PPI supported by the GIC.

Configuration

A copy of these registers is provided for each Redistributor.

These registers are configured as follows:

- GICR_IPRIORITYR0-GICR_IPRIORITYR3 store the priority of SGIs.
- GICR_IPRIORITYR4-GICR_IPRIORITYR7 store the priority of PPIs.

Attributes

GICR_IPRIORITYR<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
Priority_offset_3B								Priority_offset_2B								Priority_offset_1B								Priority_offset_0B							

Priority_offset_3B, bits [31:24]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 3. Lower priority values correspond to greater priority of the interrupt.

The reset behavior of this field is:

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

Priority_offset_2B, bits [23:16]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 2. Lower priority values correspond to greater priority of the interrupt.

The reset behavior of this field is:

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

Priority_offset_1B, bits [15:8]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 1. Lower priority values correspond to greater priority of the interrupt.

The reset behavior of this field is:

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

(old)

htmldiff from-

(new)

GICR_IPRIORITYR<n>E, Interrupt Priority Registers (extended PPI range), n = 8 - 23

The GICR_IPRIORITYR<n>E characteristics are:

Purpose

Holds the priority of the corresponding interrupt for each extended PPI supported by the GIC.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICR_IPRIORITYR<n>E are RES0.

A copy of this register is provided for each Redistributor.

Attributes

GICR_IPRIORITYR<n>E 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
Priority_offset_3B								Priority_offset_2B								Priority_offset_1B								Priority_offset_0B							

Priority_offset_3B, bits [31:24]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 3. Lower priority values correspond to greater priority of the interrupt.

The reset behavior of this field is:

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

Priority_offset_2B, bits [23:16]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 2. Lower priority values correspond to greater priority of the interrupt.

The reset behavior of this field is:

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

Priority_offset_1B, bits [15:8]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 1. Lower priority values correspond to greater priority of the interrupt.

The reset behavior of this field is:

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

The GICR_ISACTIVER0 characteristics are:

Purpose

Activates the corresponding SGI or PPI. These registers are used when saving and restoring GIC state.

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_ISACTIVER0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25
Set active bit31	Set active bit30	Set active bit29	Set active bit28	Set active bit27	Set active bit26	Set active bit25

Set_active_bit<x>, bit [x], for x = 31 to 0

Adds the active state to interrupt number x. Reads and writes have the following behavior:

Set_active_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not active, and is not active and pending. If written, has no effect.
0b1	If read, indicates that the corresponding interrupt is active, or is active and pending. If written, activates the corresponding interrupt, if the interrupt is not already active. If the interrupt is already active, the write has no effect. After a write of 1 to this bit, a subsequent read of this bit returns 1.

The reset behavior of this field is:

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

Accessing GICR_ISACTIVER0

When affinity routing is not enabled for the Security state of an interrupt in GICR_ISACTIVER0, the corresponding bit is RAZ/WI and equivalent functionality is provided by [GICD_ISACTIVER<n>](#) with n=0.

This register only applies to SGIs (bits [15:0]) and PPIs (bits [31:16]). For SPIs, this functionality is provided by [GICD_ISACTIVER<n>](#).

When [GICD_CTLR.DS](#) == 0, bits corresponding to Secure SGIs and PPIs are RAZ/WI to Non-secure accesses.

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

30/09/2021 14:53:37; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ca

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

(old)

htmldiff from-

(new)

GICR_ISACTIVER<n>E, Interrupt Set-Active Registers, n = 1 - 2

The GICR_ISACTIVER<n>E characteristics are:

Purpose

Adds the active state to the corresponding PPI.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICR_ISACTIVER<n>E are RES0.

A copy of this register is provided for each Redistributor.

Attributes

GICR_ISACTIVER<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25
Set_active_bit31	Set_active_bit30	Set_active_bit29	Set_active_bit28	Set_active_bit27	Set_active_bit26	Set_active_bit25

Set_active_bit<x>, bit [x], for x = 31 to 0

For the extended PPIs, adds the active state to interrupt number x. Reads and writes have the following behavior:

Set_active_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not active, and is not active and pending. If written, has no effect.
0b1	If read, indicates that the corresponding interrupt is active, or active and pending on this PE. If written, activates the corresponding interrupt, if the interrupt is not already active. If the interrupt is already active, the write has no effect. After a write of 1 to this bit, a subsequent read of this bit returns 1.

The reset behavior of this field is:

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

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICR_ISACTIVER<n>E number, n, is given by $n = (m-1024) \text{ DIV } 32$.
- The offset of the required GICR_ISACTIVER<n>E is $(0x200 + (4*n))$.
- The bit number of the required group modifier bit in this register is $(m-1024) \text{ MOD } 32$.

Accessing GICR_ISACTIVER<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICR_ISACTIVER<n>E, the corresponding bit is RES0.

When [GICD_CTLR](#).DS==0, bits corresponding to Secure PPIs are RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICR_ISACTIVER<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	SGI_base	0x0300 + (4 * n)	GICR_ISACTIVER<n>E

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RW**.

30/09/2021 14:53:53; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

GIC Redistributor	SGI_base	0x0100	GICR_ISENABLER0
----------------------	----------	--------	-----------------

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses This on interface this interface accessible are as follows: **RW**.

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

GICR_ISENABLER<n>E, Interrupt Set-Enable Registers, n = 1 - 2

The GICR_ISENABLER<n>E characteristics are:

Purpose

Enables forwarding of the corresponding PPI to the CPU interfaces.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICR_ISENABLER<n>E are RES0.

A copy of this register is provided for each Redistributor.

Attributes

GICR_ISENABLER<n>E is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25
Set_enable_bit31	Set_enable_bit30	Set_enable_bit29	Set_enable_bit28	Set_enable_bit27	Set_enable_bit26	Set_enable_bit25

Set_enable_bit<x>, bit [x], for x = 31 to 0

For the extended PPI range, controls the forwarding of interrupt number x to the CPU interface. Reads and writes have the following behavior:

Set_enable_bit<x>	Meaning
0b0	If read, indicates that forwarding of the corresponding interrupt is disabled. If written, has no effect.
0b1	If read, indicates that forwarding of the corresponding interrupt is enabled. If written, enables forwarding of the corresponding interrupt. After a write of 1 to this bit, a subsequent read of this bit returns 1.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICR_ISENABLER<n>E number, n, is given by $n = (m-1024) \text{ DIV } 32$.
- The offset of the required GICR_ISENABLER<n>E is $(0 \times 100 + (4 \times n))$.
- The bit number of the required group modifier bit in this register is $(m-1024) \text{ MOD } 32$.

Accessing GICR_ISENABLER<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICR_ISENABLER<n>E, the corresponding bit is RES0.

GICR_ISPENDR0, Interrupt Set-Pending Register 0

The GICR_ISPENDR0 characteristics are:

Purpose

Adds the pending state to the corresponding SGI or PPI.

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_ISPENDR0 is a 32-bit register.

Field descriptions

31	30	29	28	27	26	Se
Set_pending_bit31	Set_pending_bit30	Set_pending_bit29	Set_pending_bit28	Set_pending_bit27	Set_pending_bit26	Se

Set_pending_bit<x>, bit [x], for x = 31 to 0

For PPIs and SGIs, adds the pending state to interrupt number x. Reads and writes have the following behavior:

Set_pending_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not pending on this PE. If written, has no effect.
0b1	If read, indicates that the corresponding interrupt is pending, or active and pending on this PE. If written, changes the state of the corresponding interrupt from inactive to pending, or from active to active and pending. This has no effect in the following cases: <ul style="list-style-type: none">If the interrupt is already pending because of a write to GICR_ISPENDR0.If the interrupt is already pending because the corresponding interrupt signal is asserted. In this case, the interrupt remains pending if the interrupt signal is deasserted.

The reset behavior of this field is:

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

Accessing GICR_ISPENDR0

When affinity routing is not enabled for the Security state of an interrupt in GICR_ISPENDR0, the corresponding bit is RAZ/WI and equivalent functionality is provided by [GICD_ISPENDR<n>](#) with n=0.

This register only applies to SGIs (bits [15:0]) and PPIs (bits [31:16]). For SPIs, this functionality is provided by [GICD_ISPENDR<n>](#).

When [GICD_CTLR](#).DS == 0, bits corresponding to Secure SGIs and PPIs are RAZ/WI to Non-secure accesses.

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

30/09/2021 14:53:37; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ca

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

(old)

htmldiff from-

(new)

GICR_ISPENDR<n>E, Interrupt Set-Pending Registers, n = 1 - 2

The GICR_ISPENDR<n>E characteristics are:

Purpose

Adds the pending state to the corresponding PPI.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICR_ISPENDR<n>E are RES0.

A copy of this register is provided for each Redistributor.

Attributes

GICR_ISPENDR<n>E 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
Set_pending_bit31	Set_pending_bit30	Set_pending_bit29	Set_pending_bit28	Set_pending_bit27	Set_pending_bit26	Set_pending_bit25	Set_pending_bit24	Set_pending_bit23	Set_pending_bit22	Set_pending_bit21	Set_pending_bit20	Set_pending_bit19	Set_pending_bit18	Set_pending_bit17	Set_pending_bit16	Set_pending_bit15	Set_pending_bit14	Set_pending_bit13	Set_pending_bit12	Set_pending_bit11	Set_pending_bit10	Set_pending_bit9	Set_pending_bit8	Set_pending_bit7	Set_pending_bit6	Set_pending_bit5	Set_pending_bit4	Set_pending_bit3	Set_pending_bit2	Set_pending_bit1	Set_pending_bit0

Set_pending_bit<x>, bit [x], for x = 31 to 0

For the extended PPIs, adds the pending state to interrupt number x. Reads and writes have the following behavior:

Set_pending_bit<x>	Meaning
0b0	If read, indicates that the corresponding interrupt is not pending on this PE. If written, has no effect.
0b1	If read, indicates that the corresponding interrupt is pending, or active and pending on this PE. If written, changes the state of the corresponding interrupt from inactive to pending, or from active to active and pending. This has no effect in the following cases: <ul style="list-style-type: none"> If the interrupt is already pending because of a write to GICR_ISPENDR<n>E. If the interrupt is already pending because the corresponding interrupt signal is asserted. In this case, the interrupt remains pending if the interrupt signal is deasserted.

The reset behavior of this field is:

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

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICR_ISPENDR<n>E number, n, is given by $n = (m-1024) \text{ DIV } 32$.

- The offset of the required GICR_ISPENDR<n>E is (0x200 + (4*n)).
- The bit number of the required group modifier bit in this register is (m-1024) MOD 32.

Accessing GICR_ISPENDR<n>E

When affinity routing is not enabled for the Security state of an interrupt in GICR_ISPENDR<n>E, the corresponding bit is RES0.

When `GICD_CTLR.DS==0`, bits corresponding to Secure PPIs are RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

GICR_ISPENDR<n>E can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	SGI_base	0x0200 + (4 * n)	GICR_ISPENDR<n>E

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICR_MPAMIDR, Report maximum PARTID and PMG Register

The GICR_MPAMIDR characteristics are:

Purpose

Reports the maximum support PARTID and PMG values.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICR_MPAMIDR are RES0.

A copy of this register is provided for each Redistributor.

When [GICR_TYPER](#).MPAM==0, this register is RES0.

Attributes

GICR_MPAMIDR 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								PMGmax								PARTIDmax															

Bits [31:24]

Reserved, RES0.

PMGmax, bits [23:16]

Maximum PMG value supported.

PARTIDmax, bits [15:0]

Maximum PARTID value supported.

Accessing GICR_MPAMIDR

GICR_MPAMIDR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	RD_base	0x0018	GICR_MPAMIDR

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

Accesses This on interface this is interface accessible areas follows: **RO**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICR_NSACR, Non-secure Access Control Register

The GICR_NSACR characteristics are:

Purpose

Enables Secure software to permit Non-secure software to create SGIs targeting the PE connected to this Redistributor by writing to [ICC_SGI1R_EL1](#), [ICC_ASGI1R_EL1](#) or [ICC_SGI0R_EL1](#).

For more information, see 'Forwarding an SGI to a target PE' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Configuration

For a description on when a write to [ICC_SGI0R_EL1](#), [ICC_SGI1R_EL1](#) or [ICC_ASGI1R_EL1](#) is permitted to generate an interrupt, see 'Use of control registers for SGI forwarding' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Attributes

GICR_NSACR 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
NS_access15	NS_access14	NS_access13	NS_access12	NS_access11	NS_access10	NS_access9	NS_access8	NS_access7									

NS_access<x>, bits [2x+1:2x], for x = 15 to 0

Configures the level of Non-secure access permitted when the SGI is in Secure Group 0 or Secure Group 1, as defined from [GICR_IGROUPRO](#) and [GICR_IGRPMODR0](#). A field is provided for each SGI. The possible values of each 2-bit field are:

NS_access<x>	Meaning
0b00	Non-secure writes are not permitted to generate Secure Group 0 SGIs or Secure Group 1 SGIs.
0b01	Non-secure writes are permitted to generate a Secure Group 0 SGI.
0b10	As 0b01, but additionally Non-secure writes to are permitted to generate a Secure Group 1 SGI.
0b11	Reserved. If the field is programmed to the reserved value, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the valid values. However, to maintain the principle that as the value increases additional accesses are permitted Arm strongly recommends that implementations treat this value as 0b10. It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the valid value chosen.

The reset behavior of this field is:

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

(old)

htmldiff from-

(new)

GICR_PARTIDR, Set PARTID and PMG Register

The GICR_PARTIDR characteristics are:

Purpose

Sets the PARTID and PMG values used for memory accesses by the Redistributor.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GICR_PARTIDR are RES0.

A copy of this register is provided for each Redistributor.

When [GICR_TYPER](#).MPAM==0, this register is RES0.

Attributes

GICR_PARTIDR 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								PARTID															

Bits [31:24]

Reserved, RES0.

PMG, bits [23:16]

PMG value used when Redistributor accesses memory.

It is IMPLEMENTATION DEFINED whether bits not needed to represent PMG values in the range 0 to PMG_MAX are stateful or RES0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

PARTID, bits [15:0]

PARTID value used when Redistributor accesses memory.

It is IMPLEMENTATION DEFINED whether bits not needed to represent PARTID values in the range 0 to PARTID_MAX are stateful or RES0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

(old)htmldiff from-(new)

GICR_PENDBASER, Redistributor LPI Pending Table Base Address Register

The GICR_PENDBASER characteristics are:

Purpose

Specifies the base address of the LPI Pending table, and the Shareability and Cacheability of accesses to the LPI Pending table.

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_PENDBASER is 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	PTZ	RES0	OuterCache			RES0	Physical Address																									
Physical Address															RES0	Shareability		InnerCache			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	

Bit [63]

Reserved, RES0.

PTZ, bit [62]

Pending Table Zero. Indicates to the Redistributor whether the LPI Pending table is zero when [GICR_CTLR.EnableLPIs](#) == 1.

This field is WO, and reads as 0.

PTZ	Meaning
0b0	The LPI Pending table is not zero, and contains live data.
0b1	The LPI Pending table is zero. Software must ensure the LPI Pending table is zero before this value is written.

Bits [61:59]

Reserved, RES0.

OuterCache, bits [58:56]

Indicates the Outer Cacheability attributes of accesses to the LPI Pending table. **The possible values of this field are:**

OuterCache	Meaning
0b000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
0b001	Normal Outer Non-cacheable.
0b010	Normal Outer Cacheable Read-allocate, Write-through.
0b011	Normal Outer Cacheable Read-allocate, Write-back.
0b100	Normal Outer Cacheable Write-allocate, Write-through.
0b101	Normal Outer Cacheable Write-allocate, Write-back.
0b110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The reset behavior of this field is:

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

Bits [55:52]

Reserved, RES0.

Physical_Address, bits [51:16]

Bits [51:16] of the physical address containing the LPI Pending table.

In implementations supporting fewer than 52 bits of physical address, unimplemented upper bits are RES0.

The reset behavior of this field is:

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

Bits [15:12]

Reserved, RES0.

Shareability, bits [11:10]

Indicates the Shareability attributes of accesses to the LPI Pending table. The possible values of this field are:

Shareability	Meaning
0b00	Non-shareable.
0b01	Inner Shareable.
0b10	Outer Shareable.
0b11	Reserved. Treated as 0b00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The reset behavior of this field is:

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

InnerCache, bits [9:7]

Indicates the Inner Cacheability attributes of accesses to the LPI Pending table. The possible values of this field are:

InnerCache	Meaning
0b000	Device-nGnRnE.
0b001	Normal Inner Non-cacheable.
0b010	Normal Inner Cacheable Read-allocate, Write-through.
0b011	Normal Inner Cacheable Read-allocate, Write-back.
0b100	Normal Inner Cacheable Write-allocate, Write-through.
0b101	Normal Inner Cacheable Write-allocate, Write-back.
0b110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

The reset behavior of this field is:

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

Bits [6:0]

Reserved, RES0.

Accessing GICR_PENDBASER

Having the GICR_PENDBASER OuterCache, Shareability or InnerCache fields programmed to different values on different Redistributors with [GICR_CTLR.EnableLPIs == 1](#) in the system is UNPREDICTABLE.

Changing GICR_PENDBASER with [GICR_CTLR.EnableLPIs == 1](#) is UNPREDICTABLE.

GICR_PENDBASER can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	RD_base	0x0078	GICR_PENDBASER

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RW**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICR_PROPBASER, Redistributor Properties Base Address Register

The GICR_PROPBASER characteristics are:

Purpose

Specifies the base address of the LPI Configuration table, and the Shareability and Cacheability of accesses to the LPI Configuration table.

Configuration

A copy of this register is provided for each Redistributor.

An implementation might make this register RO, for example to correspond to an LPI Configuration table in read-only memory.

Attributes

GICR_PROPBASER is 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				OuterCache				RES0				Physical_Address																			
Physical_Address																Shareability				InnerCache				RES0				IDbits			
31	30	29	28	27	26	25	24	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.

OuterCache, bits [58:56]

Indicates the Outer Cacheability attributes of accesses to the LPI Configuration table. The possible values of this field are:

OuterCache	Meaning
0b000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
0b001	Normal Outer Non-cacheable.
0b010	Normal Outer Cacheable Read-allocate, Write-through.
0b011	Normal Outer Cacheable Read-allocate, Write-back.
0b100	Normal Outer Cacheable Write-allocate, Write-through.
0b101	Normal Outer Cacheable Write-allocate, Write-back.
0b110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The reset behavior of this field is:

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

Bits [55:52]

Reserved, RES0.

Physical_Address, bits [51:12]

Bits [51:12] of the physical address containing the LPI Configuration table.

In implementations supporting fewer than 52 bits of physical address, unimplemented upper bits are RES0.

The reset behavior of this field is:

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

Shareability, bits [11:10]

Indicates the Shareability attributes of accesses to the LPI Configuration table. **The possible values of this field are:**

Shareability	Meaning
0b00	Non-shareable.
0b01	Inner Shareable.
0b10	Outer Shareable.
0b11	Reserved. Treated as 0b00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The reset behavior of this field is:

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

InnerCache, bits [9:7]

Indicates the Inner Cacheability attributes of accesses to the LPI Configuration table. **The possible values of this field are:**

InnerCache	Meaning
0b000	Device-nGnRnE.
0b001	Normal Inner Non-cacheable.
0b010	Normal Inner Cacheable Read-allocate, Write-through.
0b011	Normal Inner Cacheable Read-allocate, Write-back.
0b100	Normal Inner Cacheable Write-allocate, Write-through.
0b101	Normal Inner Cacheable Write-allocate, Write-back.
0b110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

The reset behavior of this field is:

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

Bits [6:5]

Reserved, RES0.

IDbits, bits [4:0]

The number of bits of LPI INTID supported, minus one, by the LPI Configuration table starting at Physical_Address.

If the value of this field is larger than the value of [GICD_TYPER.IDbits](#), the [GICD_TYPER.IDbits](#) value applies.

If the value of this field is less than 0b1101, indicating that the largest INTID is less than 8192 (the smallest LPI interrupt ID), the GIC will behave as if all physical LPis are out of range.

The reset behavior of this field is:

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

Accessing GICR_PROPBASER

It is IMPLEMENTATION DEFINED whether GICR_PROPBASER can be set to different values on different Redistributors. [GICR_TYPER.CommonLPIAff](#) identifies the Redistributors that must have GICR_PROPBASER set to the same values whenever [GICR_CTLR.EnableLPis](#) == 1.

Setting different values in different copies of GICR_PROPBASER on Redistributors that are required to use a common LPI Configuration table when [GICR_CTLR.EnableLPis](#) == 1 leads to UNPREDICTABLE behavior.

Other restrictions apply when a Redistributor caches information from GICR_PROPBASER. For more information, see 'LPI Configuration tables' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

GICR_PROPBASER can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	RD_base	0x0070	GICR_PROPBASER

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses This on interface this is interface accessible areas follows: **RW**.

3020/09/2021 14:53:37; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbfdb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICR_SETLPIR, Set LPI Pending Register

The GICR_SETLPIR characteristics are:

Purpose

Generates an LPI by setting the pending state of the specified LPI.

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_SETLPIR is a 64-bit register.

Field descriptions

When GICR_TYPER.DirectLPI == 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															
																pINTID															
31	30	29	28	27	26	25	24	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.

pINTID, bits [31:0]

The INTID of the physical LPI to be generated.

Note

The size of this field is IMPLEMENTATION DEFINED, and is specified by the [GICD_TYPER.IDbits](#) field. Unimplemented bits are RES0.

When GICR_TYPER.DirectLPI == 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
																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.

(old)

htmldiff from-

(new)

GICR_STATUSR, Error Reporting Status Register

The GICR_STATUSR characteristics are:

Purpose

Provides software with a mechanism to detect:

- Accesses to reserved locations.
- Writes to read-only locations.
- Reads of write-only locations.

Configuration

A copy of this register is provided for each Redistributor.

If the GIC implementation supports two Security states this register is Banked to provide Secure and Non-secure copies.

Attributes

GICR_STATUSR 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																												WROD	RWOD	WRD	RRD

Bits [31:4]

Reserved, RES0.

WROD, bit [3]

Write to an RO location.

WROD	Meaning
0b0	Normal operation.
0b1	A write to an RO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

RWOD, bit [2]

Read of a WO location.

RWOD	Meaning
0b0	Normal operation.
0b1	A read of a WO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

WRD, bit [1]

Write to a reserved location.

WRD	Meaning
0b0	Normal operation.
0b1	A write to a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

RRD, bit [0]

Read of a reserved location.

RRD	Meaning
0b0	Normal operation.
0b1	A read of a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

Accessing GICR_STATUSR

This is an optional register. If the register is not implemented, the location is RAZ/WI.

GICR_STATUSR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	RD_base	0x0010	GICR_STATUSR (S)

This interface is accessible as follows:

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When FEAT_RME is implemented and IsAccessRoot() accesses to this register are **RW**.

Component	Frame	Offset	Instance
GIC Redistributor	RD_base	0x0010	GICR_STATUSR (NS)

This interface is accessible as follows:

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.
- When FEAT_RME is implemented and IsAccessRealm() accesses to this register are **RW**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICR_SYNCNCR, Redistributor Synchronize Register

The GICR_SYNCNCR characteristics are:

Purpose

Indicates completion of register based invalidate operations.

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_SYNCNCR 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																															Busy

Bits [31:1]

Reserved, RES0.

Busy, bit [0]

Indicates completion of invalidation operations

Busy	Meaning
0b0	No operations are in progress.
0b1	A write is in progress to one or more of the following registers: <ul style="list-style-type: none"> GICR_INVLPIR. GICR_INVALLR. GICv3, GICR_CLRLPIR.

This field tracks operations initiated on the same Redistributor.

Accessing GICR_SYNCNCR

When this register is accessed, it is optional that an implementation might wait until all operations are complete before returning a value, in which case GICR_SYNCNCR.Busy is always 0.

This register is mandatory when any of the following are true:

- [GICR_TYPER](#).Direct is 1.
- [GICR_CTLR](#).IR is 1.
- GICv4.1 is implemented.

Otherwise, the functionality is IMPLEMENTATION DEFINED.

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

30/09/2021 14:52:37; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

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

(old)

htmldiff from-

(new)

GICR_TYPER, Redistributor Type Register

The GICR_TYPER characteristics are:

Purpose

Provides information about the configuration of this Redistributor.

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_TYPER is 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
Affinity_Value																															
PPInum				VSGI				CommonLPIAff				Processor_Number				RVPEID				MPAM				DPGS				Last			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Affinity_Value, bits [63:32]

- The identity of the PE associated with this Redistributor.
- Bits [63:56] provide Aff3, the Affinity level 3 value for the Redistributor.
- Bits [55:48] provide Aff2, the Affinity level 2 value for the Redistributor.
- Bits [47:40] provide Aff1, the Affinity level 1 value for the Redistributor.
- Bits [39:32] provide Aff0, the Affinity level 0 value for the Redistributor.

PPInum, bits [31:27] When FEAT_GICv3p1 is implemented:

The value derived from this field specifies the maximum PPI INTID that a GIC implementation can support. An implementation might not implement all PPIs up to this maximum.

PPInum	Meaning
0b00000	Maximum PPI INTID is 31.
0b00001	Maximum PPI INTID is 1087.
0b00010	Maximum PPI INTID is 1119.

All other values are reserved.

Otherwise:

Reserved, RES0.

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]

The affinity level at which Redistributors share an LPI Configuration table.

CommonLPIAff	Meaning
0b00	All Redistributors must share an LPI Configuration table.
0b01	All Redistributors with the same Aff3 value must share an LPI Configuration table.
0b10	All Redistributors with the same Aff3.Aff2 value must share an LPI Configuration table.
0b11	All Redistributors with the same Aff3.Aff2.Aff1 value must share an LPI 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_INVLPPIR , 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_INVLPPIR , 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.

VLPIS, bit [1]

Indicates whether the GIC implementation supports virtual LPIs and the direct injection of virtual LPIs.

VLPIS	Meaning
0b0	The implementation does not support virtual LPIs or the direct injection of virtual LPIs.
0b1	The implementation supports virtual LPIs and the direct injection of virtual LPIs.

Note

In GICv3 implementations this field is RES0.

PLPIS, bit [0]

Indicates whether the GIC implementation supports physical LPIs.

PLPIS	Meaning
0b0	The implementation does not support physical LPIs.
0b1	The implementation supports physical LPIs.

Accessing GICR_TYPER

GICR_TYPER can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	RD_base	0x0008	GICR_TYPER

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

Accesses This on interface this is interface accessible areas follows: **RO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICR_VPENDBASER, Virtual Redistributor LPI Pending Table Base Address Register

The GICR_VPENDBASER characteristics are:

Purpose

Specifies the base address of the memory that holds the virtual LPI Pending table for the currently scheduled virtual machine.

Configuration

Attributes

GICR_VPENDBASER is a 64-bit register.

Field descriptions

When FEAT_GICv4 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
Valid	IDAI	PendingLast	Dirty	RES0	OuterCache	RES0	Physical Address																								
Physical Address															RES0	Shareability	InnerCache	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, bit [63]

This bit controls whether the virtual LPI Pending table is valid.

Valid	Meaning
0b0	The virtual LPI Pending table is not valid. No vPE is scheduled.
0b1	The virtual LPI Pending table is valid. A vPE is scheduled.

Setting GICR_VPENDBASER.Valid == 1 when the associated CPU interface does not implement FEAT_GICv4 is UNPREDICTABLE.

Note

Software can determine whether a PE supports FEAT_GICv3 or FEAT_GICv4 by reading ID_AA64PFR0_EL1.

Writing a new value to any bit of GICR_VPENDBASER, other than GICR_VPENDBASER.Valid, when GICR_VPENDBASER.Valid==1 is UNPREDICTABLE.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

IDAI, bit [62]

Implementation Defined Area Invalid. Indicates whether the IMPLEMENTATION DEFINED area in the virtual LPI Pending table is valid.

IDAI	Meaning
0b0	The IMPLEMENTATION DEFINED area is valid.
0b1	The IMPLEMENTATION DEFINED area is invalid and all pending interrupt information is held in the architecturally defined part of the virtual LPI Pending table.

For more information, see 'LPI Pending tables' and 'Virtual LPI Configuration tables and virtual LPI Pending tables' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

The reset behavior of this field is:

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

PendingLast, bit [61]

Indicates whether there are pending and enabled interrupts for the last scheduled vPE.

This value is set by the implementation when GICR_VPENDBASER.Valid has been written from 1 to 0 and is otherwise UNKNOWN.

PendingLast	Meaning
0b0	There are no pending and enabled interrupts for the last scheduled vPE.
0b1	There is at least one pending interrupt for the last scheduled vPE. It is IMPLEMENTATION DEFINED whether this bit is set when the only pending interrupts for the last scheduled vPE are not enabled. Arm deprecates setting PendingLast to 1 when the only pending interrupts for the last scheduled virtual machine are not enabled.

When the GICR_VPENDBASER.Valid bit is written from 0 to 1, this bit is RES1.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Dirty, bit [60]

When GICR_VPENDBASER.Valid == 0:

Indicates whether a de-scheduling operation is in progress.

This field is read-only.

Dirty	Meaning
0b0	No de-scheduling operation in process.
0b1	De-scheduling operation in process.

Writing 1 to GICR_VPENDBASER.Valid is UNPREDICTABLE while GICR_VPENDBASER.Dirty==1.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

When GICR_VPENDBASER.Valid == 1 and GICR_TYPER.Dirty == 1:

This field is read-only. Reports whether the Virtual Pending table has been parsed.

Dirty	Meaning
0b0	Parsing of the Virtual Pending Table has completed.
0b1	Parsing of the Virtual Pending Table has not completed.

Writing 1 to GICR_VPENDBASER.Valid is UNPREDICTABLE while GICR_VPENDBASER.Dirty == 1.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Otherwise:

This field is read-only. This field is UNKNOWN.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Bit [59]

Reserved, RES0.

OuterCache, bits [58:56]

Indicates the Outer Cacheability attributes of accesses to virtual LPI Pending tables of vPEs targeting this Redistributor.

OuterCache	Meaning
0b000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
0b001	Normal Outer Non-cacheable.
0b010	Normal Outer Cacheable Read-allocate, Write-through.
0b011	Normal Outer Cacheable Read-allocate, Write-back.
0b100	Normal Outer Cacheable Write-allocate, Write-through.
0b101	Normal Outer Cacheable Write-allocate, Write-back.
0b110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The Cacheability, Outer Cacheability and Shareability fields are used for accesses to the virtual LPI Pending table of resident and non-resident vPEs.

If the OuterCacheability attribute of the virtual LPI Pending tables that are associated with vPEs targeting the same Redistributor are different, behavior is UNPREDICTABLE.

The reset behavior of this field is:

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

Bits [55:52]

Reserved, RES0.

Physical_Address, bits [51:16]

Bits [51:16] of the physical address containing the virtual LPI Pending table.

In implementations supporting fewer than 52 bits of physical address, unimplemented upper bits are RES0.

The reset behavior of this field is:

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

Bits [15:12]

Reserved, RES0.

Shareability, bits [11:10]

Indicates the Shareability attributes of accesses to the virtual LPI Pending table.

Shareability	Meaning
0b00	Non-shareable.
0b01	Inner Shareable.
0b10	Outer Shareable.
0b11	Reserved. Treated as 0b00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The Cacheability, Outer Cacheability and Shareability fields are used for accesses to the virtual LPI Pending table of resident and non-resident vPEs.

If the Shareability attribute of the virtual LPI Pending tables that are associated with vPEs targeting the same Redistributor are different, behavior is UNPREDICTABLE.

The reset behavior of this field is:

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

InnerCache, bits [9:7]

Indicates the Inner Cacheability attributes of accesses to the virtual LPI Pending table.

InnerCache	Meaning
0b000	Device-nGnRnE.
0b001	Normal Inner Non-cacheable.
0b010	Normal Inner Cacheable Read-allocate, Write-through.
0b011	Normal Inner Cacheable Read-allocate, Write-back.
0b100	Normal Inner Cacheable Write-allocate, Write-through.
0b101	Normal Inner Cacheable Write-allocate, Write-back.
0b110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

The Cacheability, Outer Cacheability and Shareability fields are used for accesses to the virtual LPI Pending table of resident and non-resident vPEs.

If the InnerCacheability attribute of the virtual LPI Pending tables that are associated with vPEs targeting the same Redistributor are different, behavior is UNPREDICTABLE.

The reset behavior of this field is:

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

Bits [6:0]

Reserved, RES0.

When FEAT_GICv4p1 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	
Valid	Doorbell	PendingLast	Dirty	VGrp0En	VGrp1En	RES0																										
RES0																	vPEID															
31	30	29	28	27	26	25	24	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, bit [63]

This bit controls whether a vPE is scheduled:

Valid	Meaning
0b0	The virtual LPI Pending table is not valid. No vPE is scheduled.
0b1	The virtual LPI Pending table is valid. A vPE is scheduled.

Setting GICR_VPENDBASER.Valid == 1 when the associated CPU interface does not implement FEAT_GICv4 is UNPREDICTABLE.

Note

Software can determine whether a PE supports FEAT_GICv3 or FEAT_GICv4 by reading ID_AA64PFR0_EL1.

Writing a new value to any bit of GICR_VPENDBASER, other than GICR_VPENDBASER.Valid, when GICR_VPENDBASER.Valid==1 is UNPREDICTABLE.

Setting GICR_VPENDBASER.Valid to 1 is UNPREDICTABLE if [GICR_VPROPBASER](#).Valid == 0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Doorbell, bit [62]

When GICR_VPENDBASER.Valid is written from 1 to 0, this bit controls whether a default doorbell interrupt is requested for the descheduled vPE.

Doorbell	Meaning
0b0	No default doorbell requested.
0b1	Default doorbell requested.

When GICR_VPENDBASER.Valid is written from 1 to 0, if there are outstanding enabled pending interrupts then this bit is treated as 0.

When GICR_VPENDBASER.Valid is written from 1 to 0, if GICR_VPENDBASER.PendingLast is written as 1 then this bit is treated as 0.

When GICR_VPENDBASER.Valid == 1, reads return an UNKNOWN value.

The reset behavior of this field is:

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

PendingLast, bit [61]

Indicates whether there are pending and enabled interrupts for the last scheduled vPE.

This value is set by the implementation when GICR_VPENDBASER.Valid is written from 1 to 0 and is otherwise UNKNOWN.

PendingLast	Meaning
0b0	There are no pending and enabled interrupts for the last scheduled vPE.
0b1	There is at least one pending and enabled interrupt for the last scheduled vPE.

When the GICR_VPENDBASER.Valid bit is written from 0 to 1, this bit is RES1.

When GICR_VPENDBASER.Valid is written from 1 to 0, if GICR_VPENDBASER.PendingLast is written as 1, then this bit is set to an UNKNOWN value.

The reset behavior of this field is:

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

Dirty, bit [60]**When GICR_VPENDBASER.Valid == 0:**

Read-only. Indicates whether a de-scheduling operation is in progress.

Dirty	Meaning
0b0	No de-scheduling operation in progress.
0b1	De-scheduling operation in progress.

Writing 1 to GICR_VPENDBASER.Valid is UNPREDICTABLE while GICR_VPENDBASER.Dirty == 1.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Otherwise:

Read-only. Reports whether the Virtual Pending table has been parsed.

Dirty	Meaning
0b0	Parsing of the Virtual Pending Table is complete.
0b1	Parsing of the Virtual Pending Table has not completed.

Writing 1 to GICR_VPENDBASER.Valid is UNPREDICTABLE while GICR_VPENDBASER.Dirty == 1.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

VGrp0En, bit [59]

Enable virtual Group 0 interrupts.

VGrp0En	Meaning
0b0	Forwarding of virtual Group 0 interrupts disabled.
0b1	Forwarding of virtual Group 0 interrupts enabled.

Writing a new value to VGrp0En while [GICR_VPENDBASER.Valid==1](#) is CONSTRAINED UNPREDICTABLE:

- The update is ignored.
- The update is ignored for all purposes other than a direct read of the register.
- The virtual group enable is updated.

The reset behavior of this field is:

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

VGrp1En, bit [58]

Enable virtual Group 1 interrupts.

VGrp1En	Meaning
0b0	Forwarding of virtual Group 1 interrupts disabled.
0b1	Forwarding of virtual Group 1 interrupts enabled.

Writing a new value to VGrp1En while [GICR_VPENDBASER.Valid==1](#) is CONSTRAINED UNPREDICTABLE:

- The update is ignored.
- The update is ignored for all purposes other than a direct read of the register.
- The virtual group enable is updated.

The reset behavior of this field is:

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

Bits [57:16]

Reserved, RES0.

vPEID, bits [15:0]

When GICR_VPENDBASER.Valid == 1, ID of scheduled vPE.

When GICR_VPENDBASER.Valid == 1, if GICR_VPENDBASER.vPEID is set to a value greater than the configured vPEID width, the behavior of this field is CONSTRAINED UNPREDICTABLE:

- GICR_VPENDBASER.vPEID is treated as having an UNKNOWN valid value for all purposes other than a direct read of the register.
- GICR_VPENDBASER.Valid is treated as being set to 0 for all purposes other than a direct read of the register.

The size of this field is IMPLEMENTATION DEFINED, and is specified by the GICD_TYPER2.VIL and GICD_TYPER2.VID fields, unimplemented bits are RES0.

Accessing GICR_VPENDBASER

The effect of a write to this register is not guaranteed to be visible throughout the affinity hierarchy, as indicated by GICR_CTLR.RWP == 0.

GICR_VPENDBASER can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	VLPI_base	0x0078	GICR_VPENDBASER

- When GICD_CTLR.DS == 0 accesses to this register are RW.
- When an access is Secure accesses to this register are RW.
- When an access is Non-secure accesses to this register are RW.

AccessesThis oninterface thisis interfaceaccessible areas follows: RW.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbb36e47856e443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

The GICR_VPROPBASER characteristics are:

Purpose

Specifies the base address of the memory that holds the virtual LPI Configuration table for the currently scheduled virtual machine.

Configuration

This register is provided in FEAT_GICv4 implementations only.

Attributes

GICR_VPROPBASER is a 64-bit register.

Field descriptions

When FEAT_GICv4 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				OuterCache				RES0				Physical_Address																			
Physical_Address												Shareability				InnerCache				RES0		IDbits									
31	30	29	28	27	26	25	24	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.

OuterCache, bits [58:56]

Indicates the Outer Cacheability attributes of accesses to the LPI Configuration table. The possible values of this field are:

OuterCache	Meaning
0b000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
0b001	Normal Outer Non-cacheable.
0b010	Normal Outer Cacheable Read-allocate, Write-through.
0b011	Normal Outer Cacheable Read-allocate, Write-back.
0b100	Normal Outer Cacheable Write-allocate, Write-through.
0b101	Normal Outer Cacheable Write-allocate, Write-back.
0b110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The reset behavior of this field is:

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

Bits [55:52]

Reserved, RES0.

Physical_Address, bits [51:12]

Bits [51:12] of the physical address containing the virtual LPI Configuration table.

In implementations supporting fewer than 52 bits of physical address, unimplemented upper bits are RES0.

The reset behavior of this field is:

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

Shareability, bits [11:10]

Indicates the Shareability attributes of accesses to the LPI Configuration table. **The possible values of this field are:**

Shareability	Meaning
0b00	Non-shareable.
0b01	Inner Shareable.
0b10	Outer Shareable.
0b11	Reserved. Treated as 0b00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The reset behavior of this field is:

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

InnerCache, bits [9:7]

Indicates the Inner Cacheability attributes of accesses to the LPI Configuration table. **The possible values of this field are:**

InnerCache	Meaning
0b000	Device-nGnRnE.
0b001	Normal Inner Non-cacheable.
0b010	Normal Inner Cacheable Read-allocate, Write-through.
0b011	Normal Inner Cacheable Read-allocate, Write-back.
0b100	Normal Inner Cacheable Write-allocate, Write-through.
0b101	Normal Inner Cacheable Write-allocate, Write-back.
0b110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

The reset behavior of this field is:

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

Bits [6:5]

Reserved, RES0.

IDbits, bits [4:0]

The number of bits of virtual LPI INTID supported, minus one.

If the value of this field is less than 0b1101, indicating that the largest INTID is less than 8192 (the smallest LPI interrupt ID), the GIC will behave as if all virtual LPIs are out of range.

The reset behavior of this field is:

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

When FEAT_GICv4p1 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		
Valid	RES0	Entry_Size				OuterCache				Indirect	Page_Size				Z	Physical Address																	
											Physical Address											Shareability				InnerCache				Size			
31	30	29	28	27	26	25	24	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, bit [63]

This bit controls whether the vPE Configuration Table is valid.

Valid	Meaning
0b0	The vPE Configuration table is not valid.
0b1	The vPE Configuration table is valid.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Bit [62]

Reserved, RES0.

Entry_Size, bits [61:59]

Specifies the number 64-bit doublewords per table entry, minus one.

This bit is read-only.

OuterCache, bits [58:56]

Indicates the Outer Cacheability attributes of accesses to the table. **The possible values of this field are:**

OuterCache	Meaning
0b000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
0b001	Normal Outer Non-cacheable.
0b010	Normal Outer Cacheable Read-allocate, Write-through.
0b011	Normal Outer Cacheable Read-allocate, Write-back.
0b100	Normal Outer Cacheable Write-allocate, Write-through.
0b101	Normal Outer Cacheable Write-allocate, Write-back.
0b110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The reset behavior of this field is:

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

Indirect, bit [55]

This field indicates whether GICR_VPROPBASER specifies a single, flat table or a two-level table where the first level contains a list of descriptors.

Indirect	Meaning
0b0	Single Level. The Size field indicates the number of pages used to store data associated with each table entry.
0b1	Two Level. The Size field indicates the number of pages that contain an array of 64-bit descriptors to pages that are used to store the data associated with each table entry. A little endian memory order model is used.

This field is RAZ/WI for GIC implementations that only support flat tables.

This field is RES0 for GIC implementations that only support flat tables.

If the supported vPEID width indicated by [GICD_TYPER2.VIL](#) and [GICD_TYPER2.VID](#), and the smallest page size that is supported result in a single level table that requires multiple pages, then implementing this bit as RAZ/WI is deprecated.

The reset behavior of this field is:

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

Page_Size, bits [54:53]

The following values indicate the size of page that the translation table uses:

Page_Size	Meaning
0b00	4KB.
0b01	16KB.
0b10	64KB.
0b11	Reserved. Treated as 0b10.

Note

If the GIC implementation supports only a single, fixed page size, this field might be RO.

The reset behavior of this field is:

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

Z, bit [52]

When GICR_VPROPBASER.Valid is written from 0 to 1, GICR_VPROPBASER.Z indicates whether the vPE Configuration table is known to contain all zeros.

Z	Meaning
0b0	The vPE Configuration table is not zero, and contains live data.
0b1	The vPE Configuration table is zero.

Setting GICR_VPROPBASER.Z to 0 causes the IRI to reload configuration from memory

When GICR_VPROPBASER.Valid is written from 0 to 1, if GICR_VPROPBASER.Z==1 behavior is UNPREDICTABLE if the allocated memory does not contain all zeros.

This field is WO, and reads as 0.

Physical_Address, bits [51:12]

Bits [51:12] of the physical address containing the LPI Configuration table.

In implementations supporting fewer than 52 bits of physical address, unimplemented upper bits are RES0.

The reset behavior of this field is:

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

Shareability, bits [11:10]

Indicates the Shareability attributes of accesses to the LPI Configuration table. ~~The possible values of this field are:~~

Shareability	Meaning
0b00	Non-shareable.
0b01	Inner Shareable.
0b10	Outer Shareable.
0b11	Reserved. Treated as 0b00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The reset behavior of this field is:

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

InnerCache, bits [9:7]

Indicates the Inner Cacheability attributes of accesses to the LPI Configuration table. ~~The possible values of this field are:~~

InnerCache	Meaning
0b000	Device-nGnRnE.
0b001	Normal Inner Non-cacheable.
0b010	Normal Inner Cacheable Read-allocate, Write-through.
0b011	Normal Inner Cacheable Read-allocate, Write-back.
0b100	Normal Inner Cacheable Write-allocate, Write-through.
0b101	Normal Inner Cacheable Write-allocate, Write-back.
0b110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

The reset behavior of this field is:

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

Size, bits [6:0]

The number of pages of physical memory allocated to the table, minus one.

[GICR_VPROPBASER](#).Page_Size specifies the size of each page.

The reset behavior of this field is:

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

Accessing GICR_VPROPBASER

GICR_VPROPBASER can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	VLPI_base	0x0070	GICR_VPROPBASER

- ~~When GICD_CTLR.DS == 0 accesses to this register are RW.~~
- ~~When an access is Secure accesses to this register are RW.~~
- ~~When an access is Non-secure accesses to this register are RW.~~

~~AccessesThis on interface thisis interfaceaccessible areas follows: RW.~~

3020/09/2021 14:12:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

(old)

htmldiff from-

(new)

GICR_VSGIPENDR, Redistributor virtual SGI pending state register

The GICR_VSGIPENDR characteristics are:

Purpose

Requests the pending state of virtual SGIs for a specified vPE.

Configuration

This register is present only when FEAT_GICv4p1 is implemented. Otherwise, direct accesses to GICR_VSGIPENDR are RES0.

A copy of this register is provided for each Redistributor.

Attributes

GICR_VSGIPENDR 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
Busy																Pending															
RES0																															

Busy, bit [31]

ID of target vPEID

Busy	Meaning
0b0	Query of virtual SGI state not in progress.
0b1	Query of virtual SGI state in progress.

Bits [30:16]

Reserved, RES0.

Pending, bits [15:0]

Pending state of virtual SGIs for requested vPEID.

This field is UNKNOWN when [GICR_VSGIPENDR](#).Busy == 1

Accessing GICR_VSGIPENDR

64-bit access only.

GICR_VSGIPENDR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
GIC Redistributor	VLPI_base	0x0088	GICR_VSGIPENDR

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

Accesses This on interface this is interface accessible areas follows: **RO**.

3020/09/2021 14:12:5237: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GICR_VSGIR, Redistributor virtual SGI pending state request register

The GICR_VSGIR characteristics are:

Purpose

Requests the pending state of virtual SGIs for a specified vPE.

Configuration

This register is present only when FEAT_GICv4p1 is implemented. Otherwise, direct accesses to GICR_VSGIR are RES0.

A copy of this register is provided for each Redistributor.

Attributes

GICR_VSGIR 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																vPEID															

Bits [31:16]

Reserved, RES0.

vPEID, bits [15:0]

ID of target vPE

Writing this field is CONSTRAINED UNPREDICTABLE when [GICR_VSGIPENDR](#).Busy == 1, with either the write ignored or a new query started.

Writing a value greater than the configured vPEID width behaviour is CONSTRAINED UNPREDICTABLE:

- GICR_VPEINDBASER.vPEID is treated as having an UNKNOWN valid value for all purposes other than a direct read of the register.
- GICR_VPEINDBASER.Valid is treated as being set to 0 for all purposes other than a direct read of the register.

The size of this field is IMPLEMENTATION DEFINED, and is specified by the [GICD_TYPER2](#).VIL and [GICD_TYPER2](#).VID fields. Unimplemented bits are RES0.

Accessing GICR_VSGIR

64-bit access only.

GICR_VSGIR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
-----------	-------	--------	----------

GIC Redistributor	VLPI_base	0x0080	GICR_VSGIR
----------------------	-----------	--------	------------

- When GICD_CTLR.DS == 0 accesses to this register are **WO**.
- When an access is Secure accesses to this register are **WO**.
- When an access is Non-secure accesses to this register are **WO**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **WO.**

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fe44a233fbbdb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The GICR_WAKER characteristics are:

Purpose

Permits software to control the behavior of the WakeRequest power management signal corresponding to the Redistributor. Power management operations follow the rules in 'Power management' in in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Configuration

A copy of this register is provided for each Redistributor.

Attributes

GICR_WAKER is a 32-bit register.

Field descriptions

31	3029282726252423222120191817161514131211109876543	2	1	0
IMPLEMENTATION DEFINED	RES0	ChildrenAsleep	ProcessorSleep	IMPLEMENTATION DEFINED

IMPLEMENTATION DEFINED, bit [31]

IMPLEMENTATION DEFINED.

Bits [30:3]

Reserved, RES0.

ChildrenAsleep, bit [2]

Read-only. Indicates whether the connected PE is quiescent:

ChildrenAsleep	Meaning
0b0	An interface to the connected PE might be active.
0b1	All interfaces to the connected PE are quiescent.

The reset behavior of this field is:

- On a GIC reset, this field resets to 1.

ProcessorSleep, bit [1]

Indicates whether the Redistributor can assert the **WakeRequest** signal:

ProcessorSleep	Meaning
0b0	This PE is not in, and is not entering, a low power state.
0b1	<p>The PE is either in, or is in the process of entering, a low power state.</p> <p>All interrupts that arrive at the Redistributor:</p> <ul style="list-style-type: none"> Assert a WakeRequest signal. Are held in the pending state at the Redistributor, and are not communicated to the CPU interface. <hr/> <p>Note</p> <p>When ProcessorSleep == 1, the Redistributor must ensure that any interrupts that are pending on the CPU interface are released.</p> <hr/> <p>For an implementation that is using the GIC Stream Protocol Interface:</p> <ul style="list-style-type: none"> A Quiesce command puts the interface between the Redistributor and the CPU interface in a quiescent state. For more information, see 'Quiesce (IRI)' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069). A Release command releases any interrupts that are pending on the CPU interface. For more information, see 'Release (ICC)' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

Note

Before powering down a PE, software must set this bit to 1 and wait until ChildrenAsleep == 1. After powering up a PE, or following a failed powerdown, software must set this bit to 0 and wait until ChildrenAsleep == 0.

Changing ProcessorSleep from 1 to 0 when ChildrenAsleep is not 1 results in UNPREDICTABLE behavior.

Changing ProcessorSleep from 0 to 1 when the Enable for each interrupt group in the associated CPU interface is not 0 results in UNPREDICTABLE behavior.

The reset behavior of this field is:

- On a GIC reset, this field resets to 1.

IMPLEMENTATION DEFINED, bit [0]

IMPLEMENTATION DEFINED.

Accessing GICR_WAKER

When `GICD_CTLR.DS==1`, this register is always accessible.

When `GICD_CTLR.DS==0`, this is a Secure register. This register is RAZ/WI to Non-secure accesses.

To ensure a Redistributor is quiescent, software must write to GICR_WAKER with ProcessorSleep == 1, then poll the register until ChildrenAsleep == 1.

Resetting the connected PE when GICR_WAKER.ProcessorSleep==0 or GICR_WAKER.ChildrenAsleep==0, can lead to UNPREDICTABLE behaviour in the IRI.

Resetting the IRI when GICR_WAKER.ProcessorSleep==0 or GICR_WAKER.ChildrenAsleep==0 can lead to UNPREDICTABLE behaviour in the connected PE.

This interface is accessible as follows:

- 3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffdbdfb36e47856c443a7ce9a85f5e501ca

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

(old)

htmldiff from-

(new)

GITS_BASER<n>, ITS Translation Table Descriptors, n = 0 - 7

The GITS_BASER<n> characteristics are:

Purpose

Specifies the base address and size of the ITS translation tables.

Configuration

A copy of this register is provided for each ITS translation table.

Bits [63:32] and bits [31:0] are accessible independently.

A maximum of 8 GITS_BASER<n> registers can be provided. Unimplemented registers are RES0.

When [GITS_CTLR.Enabled](#) == 1 or [GITS_CTLR.Quiescent](#) == 0, writing this register is UNPREDICTABLE.

Attributes

GITS_BASER<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	
Valid	Indirect	InnerCache	Type	OuterCache	Entry_Size	Physical_Address																										
Physical_Address																Shareability				Page_Size				Size								
31	30	29	28	27	26	25	24	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, bit [63]

Indicates whether software has allocated memory for the translation table:

Valid	Meaning
0b0	No memory is allocated for the translation table. The ITS discards any writes to the interrupt translation page when either: <ul style="list-style-type: none"> GITS_BASER<n>.Type specifies any valid table entry type other than interrupt collections, that is, any value other than 0b100. GITS_BASER<n>.Type specifies an interrupt collection and GITS_TYPER.HCC == 0.
0b1	Memory is allocated to the translation table.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Indirect, bit [62]

This field indicates whether an implemented register specifies a single, flat table or a two-level table where the first level contains a list of descriptors.

Indirect	Meaning
0b0	Single Level. The Size field indicates the number of pages used by the ITS to store data associated with each table entry.
0b1	Two Level. The Size field indicates the number of pages which contain an array of 64-bit descriptors to pages that are used to store the data associated with each table entry. A little endian memory order model is used.

For more information, see 'The ITS tables' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

This field is RAZ/WI for GIC implementations that only support flat tables. If the maximum width of the scaling factor that is identified by GITS_BASER<n>.Type and the smallest page size that is supported result in a single level table that requires multiple pages, then implementing this bit as RAZ/WI is DEPRECATED.

The reset behavior of this field is:

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

InnerCache, bits [61:59]

Indicates the Inner Cacheability attributes of accesses to the table. The possible values of this field are:

InnerCache	Meaning
0b000	Device-nGnRnE.
0b001	Normal Inner Non-cacheable.
0b010	Normal Inner Cacheable Read-allocate, Write-through.
0b011	Normal Inner Cacheable Read-allocate, Write-back.
0b100	Normal Inner Cacheable Write-allocate, Write-through.
0b101	Normal Inner Cacheable Write-allocate, Write-back.
0b110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

The reset behavior of this field is:

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

Type, bits [58:56]

Read only. Specifies the type of entity that requires entries in the corresponding translation table. The possible values of the field are:

Type	Meaning
0b000	Unimplemented. This register does not correspond to a translation table.
0b001	Devices. This register corresponds to a translation table that scales with the width of the DeviceID. Only a single GITS_BASER<n> register reports this type.
0b010	vPEs. FEAT_GICv4 only. This register corresponds to a translation table that scales with the number of vPEs in the system. The translation table requires (ENTRY_SIZE * N) bytes of memory, where N is the number of vPEs in the system. Only a single GITS_BASER<n> register reports this type.
0b100	Interrupt collections. This register corresponds to a translation table that scales with the number of interrupt collections in the system. The translation table requires (ENTRY_SIZE * N) bytes of memory, where N is the number of interrupt collections. Not more than one GITS_BASER<n> register will report this type.

Other values are reserved.

For FEAT_GICv4p1, the registers are allocated as follows:

- GITS_BASER0.Type is 0b001 (Device).
- GITS_BASER1.Type is either 0b100 (Collection Table) or 0b000 (Unimplemented).
- GITS_BASER2.Type is either 0b010 (vPE) or 0b000 (Unimplemented).
- GITS_BASER<n>.Type, where 'n' is in the range 3 to 7, is 0b000 (Unimplemented).

For FEAT_GICv3, FEAT_GICv3p1, and FEAT_GICv4, Arm recommends that the GITS_BASER<n> use the same allocations.

Other allocations of Type values are deprecated.

The reset behavior of this field is:

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

OuterCache, bits [55:53]

Indicates the Outer Cacheability attributes of accesses to the table. The possible values of this field are:

OuterCache	Meaning
0b000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
0b001	Normal Outer Non-cacheable.
0b010	Normal Outer Cacheable Read-allocate, Write-through.
0b011	Normal Outer Cacheable Read-allocate, Write-back.
0b100	Normal Outer Cacheable Write-allocate, Write-through.
0b101	Normal Outer Cacheable Write-allocate, Write-back.
0b110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The reset behavior of this field is:

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

Entry_Size, bits [52:48]

Read-only. Specifies the number of bytes per translation table entry, minus one.

Physical_Address, bits [47:12]

Physical Address. When Page_Size is 4KB or 16KB:

- Bits [51:48] of the base physical address are zero.
- This field provides bits[47:12] of the base physical address of the table.
- Bits[11:0] of the base physical address are zero.
- The address must be aligned to the size specified in the Page Size field. Otherwise the effect is CONSTRAINED UNPREDICTABLE, and can be one of the following:
 - Bits[X:12], where X is derived from the page size, are treated as zero.
 - The value of bits[X:12] are used when calculating the address of a table access.

When Page_Size is 64KB:

- Bits[47:16] of the register provide bits[47:16] of the base physical address of the table.
- Bits[15:12] of the register provide bits[51:48] of the base physical address of the table.
- Bits[15:0] of the base physical address are 0.

In implementations that support fewer than 52 bits of physical address, any unimplemented upper bits might be RAZ/WI.

The reset behavior of this field is:

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

Shareability, bits [11:10]

Indicates the Shareability attributes of accesses to the table. The possible values of this field are:

Shareability	Meaning
0b00	Non-shareable.
0b01	Inner Shareable.
0b10	Outer Shareable.
0b11	Reserved. Treated as 0b00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The reset behavior of this field is:

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

Page_Size, bits [9:8]

The size of page that the translation table uses:

Page_Size	Meaning
0b00	4KB.
0b01	16KB.
0b10	64KB.
0b11	Reserved. Treated as 0b10.

Note

If the GIC implementation supports only a single, fixed page size, this field might be RO.

The reset behavior of this field is:

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

Size, bits [7:0]

The number of pages of physical memory allocated to the table, minus one. GITS_BASER<n>.Page_Size specifies the size of each page.

If GITS_BASER<n>.Type == 0, this field is RAZ/WI.

The reset behavior of this field is:

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

Accessing GITS_BASER<n>

GITS_BASER<n> can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
GIC ITS control	0x0100 + (8 * n)	GITS_BASER<n>

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

Accesses This on interface this is interface accessible areas follows: **RW**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

GITS_CBASER, ITS Command Queue Descriptor

The GITS_CBASER characteristics are:

Purpose

Specifies the base address and size of the ITS command queue.

Configuration

Bits [63:32] and bits [31:0] are accessible separately.

Attributes

GITS_CBASER is 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	
Valid	RES0	InnerCache				RES0	OuterCache				RES0											Physical_Address										
										Physical_Address										Shareability		RES0	Size									
31	30	29	28	27	26	25	24	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, bit [63]

Indicates whether software has allocated memory for the command queue:

Valid	Meaning
0b0	No memory is allocated for the command queue.
0b1	Memory is allocated to the command queue.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Bit [62]

Reserved, RES0.

InnerCache, bits [61:59]

Indicates the Inner Cacheability attributes of accesses to the command queue. The possible values of this field are:

InnerCache	Meaning
0b000	Device-nGnRnE.
0b001	Normal Inner Non-cacheable.
0b010	Normal Inner Cacheable Read-allocate, Write-through.
0b011	Normal Inner Cacheable Read-allocate, Write-back.
0b100	Normal Inner Cacheable Write-allocate, Write-through.
0b101	Normal Inner Cacheable Write-allocate, Write-back.
0b110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

The reset behavior of this field is:

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

Bits [58:56]

Reserved, RES0.

OuterCache, bits [55:53]

Indicates the Outer Cacheability attributes of accesses to the command queue. The possible values of this field are:

OuterCache	Meaning
0b000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
0b001	Normal Outer Non-cacheable.
0b010	Normal Outer Cacheable Read-allocate, Write-through.
0b011	Normal Outer Cacheable Read-allocate, Write-back.
0b100	Normal Outer Cacheable Write-allocate, Write-through.
0b101	Normal Outer Cacheable Write-allocate, Write-back.
0b110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
0b111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The reset behavior of this field is:

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

Bit [52]

Reserved, RES0.

Physical_Address, bits [51:12]

Bits [51:12] of the base physical address of the command queue. Bits [11:0] of the base address are 0.

In implementations supporting fewer than 52 bits of physical address, unimplemented upper bits are RES0.

If bits [15:12] are not all zeros, behavior is a CONSTRAINED UNPREDICTABLE choice:

- Bits [15:12] are treated as if all the bits are zero. The value read back from those bits is either the value written or zero.
- The result of the calculation of an address for a command queue read can be corrupted.

The reset behavior of this field is:

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

Shareability, bits [11:10]

Indicates the Shareability attributes of accesses to the command queue. The possible values of this field are:

Shareability	Meaning
0b00	Non-shareable.
0b01	Inner Shareable.
0b10	Outer Shareable.
0b11	Reserved. Treated as 0b00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The reset behavior of this field is:

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

Bits [9:8]

Reserved, RES0.

Size, bits [7:0]

The number of 4KB pages of physical memory allocated to the command queue, minus one.

The reset behavior of this field is:

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

The command queue is a circular buffer and wraps at Physical Address $[47:0] + (4096 * (\text{Size} + 1))$.

Note

When this register is successfully written, the value of `GITS_CREADR` is set to zero.

Accessing GITS_CBASER

When `GITS_CTLR.Enabled == 1` or `GITS_CTLR.Quiescent == 0`, writing this register is UNPREDICTABLE.

GITS_CBASER can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
GIC ITS control	0x0080	GITS CBASER

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RW**.

3020/09/2021 1412:5337: 092b4e1bbfb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GITS_CTLR, ITS Control Register

The GITS_CTLR characteristics are:

Purpose

Controls the operation of an ITS.

Configuration

The ITS_Number (bits [7:4]) and bit [1] fields apply only in FEAT_GICv4 implementations, and are RES0 in FEAT_GICv3 implementations.

Attributes

GITS_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
Quiescent											RES0												UMSlrq		ITS Number		RES0	ImDe	Enabled		

Quiescent, bit [31]

Read-only. Indicates completion of all ITS operations when GITS_CTLR.Enabled == 0.

Quiescent	Meaning
0b0	The ITS is not quiescent and cannot be powered down.
0b1	The ITS is quiescent and can be powered down.

For the ITS to be considered inactive, there must be no transactions in progress. In addition, all operations required to ensure that mapping data is consistent with external memory must be complete.

Note

In distributed GIC implementations, this bit is set to 1 only after the ITS forwards any operations that have not yet been completed to the Redistributors and receives confirmation that all such operations have reached the appropriate Redistributor.

In FEAT_GICv3, FEAT_GICv3p1, and FEAT_GICv4, when GITS_CTLR.Enabled == 1, the value of GITS_CTLR.Quiescent is UNKNOWN.

In FEAT_GICv4p1, when GITS_CTLR.Enabled == 1, the value of GITS_CTLR.Quiescent reads as 1 until the write to Enabled has taken effect and then reads as 0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 1.

Bits [30:9]

Reserved, RES0.

UMSIirq, bit [8]

Unmapped MSI reporting interrupt enable.

UMSIirq	Meaning
0b0	The ITS does not assert an interrupt signal when GITS_STATUSR.UMSI is 1.
0b1	The ITS asserts an interrupt signal when GITS_STATUSR.UMSI is 1.

If [GITS_TYPER.UMSIirq](#) is 0, this field is RES0.

The reset behavior of this field is:

- On a Warm reset, this field resets to 0.

ITS_Number, bits [7:4]

In FEAT_GICv3 implementations this field is RES0.

In FEAT_GICv4 implementations with more than one ITS instance, this field indicates the ITS number for use with 'VMOVP GICv4.0' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

It is IMPLEMENTATION DEFINED whether this field is programmable or RO.

If this field is programmable, changing this field when `GITS_CTLR.Quiescent == 0` or `GITS_CTLR.Enabled == 1` is UNPREDICTABLE.

The reset behavior of this field is:

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

Bits [3:2]

Reserved, RES0.

ImDe, bit [1]

In GICv3 implementations, this bit is RES0.

In GICv4 implementations, this bit is IMPLEMENTATION DEFINED.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Enabled, bit [0]

Controls whether the ITS is enabled:

Enabled	Meaning
0b0	The ITS is not enabled. Writes to GITS_TRANSLATER are ignored and no further command queue entries are processed.
0b1	The ITS is enabled. Writes to GITS_TRANSLATER result in interrupt translations and the command queue is processed.

If a write to this register changes this field from 1 to 0, the ITS must ensure that both:

- Any caches containing mapping data are made consistent with external memory.
- `GITS_CTLR.Quiescent == 0` until all caches are consistent with external memory.

Changing `GITS_CTLR.Enabled` from 0 to 1 when `GITS_CTLR.Quiescent` is 0 results in UNPREDICTABLE behavior.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Accessing GITS_CTLR

GITS_CTLR can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
GIC ITS control	0x0000	GITS_CTLR

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: RW.

3020/09/2021 1412:5337; 092b4e1bbfb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

GITS_CWRITER, ITS Write Register

The GITS_CWRITER characteristics are:

Purpose

Specifies the offset from [GITS_CBASER](#) where software writes the next ITS command.

Configuration

Bits [63:32] and bits [31:0] are accessible separately.

Attributes

GITS_CWRITER is 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												Offset																RES0				Retry
31	30	29	28	27	26	25	24	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.

Offset, bits [19:5]

Bits [19:5] of the offset from [GITS_CBASER](#). Bits [4:0] of the offset are zero.

The reset behavior of this field is:

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

Bits [4:1]

Reserved, RES0.

Retry, bit [0]

Writing this bit has the following effects:

Retry	Meaning
0b0	No effect on the processing commands by the ITS.
0b1	Restarts the processing of commands by the ITS if it stalled because of a command error.
Note	
If the processing of commands is not stalled because of a command error, writing 1 to this bit has no effect.	

When read, this bit is RES0.

For more information, see 'The ITS command interface' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

If `GITS_CWRITER` is written with a value outside of the valid range specified by [GITS_CBASER.Physical_Address](#) and [GITS_CBASER.Size](#), behavior is a CONSTRAINED UNPREDICTABLE choice, as follows:

- The command queue is considered invalid, and no further commands are processed until `GITS_CWRITER` is written with a value that is in the valid range.
- The value is treated as a valid `UNKNOWN` value.

An implementation might choose to report a system error in an IMPLEMENTATION DEFINED manner.

Accessing GITS_CWRITER

GITS_CWRITER can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
GIC ITS control	0x0088	GITS_CWRITER

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RW**.

3020/09/2021 1412:5237; 092b4e1bbfb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GITS_IIDR, ITS Identification Register

The GITS_IIDR characteristics are:

Purpose

Provides information about the implementer and revision of the ITS.

Configuration

This register is available in all configurations of the GIC. If the GIC implementation supports two Security states, this register is Common.

Attributes

GITS_IIDR 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
ProductID								RES0				Variant				Revision				Implementer											

ProductID, bits [31:24]

Product Identifier.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Bits [23:20]

Reserved, RES0.

Variant, bits [19:16]

Variant number. Typically, this field is used to distinguish product variants, or major revisions of a product.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Revision, bits [15:12]

Revision number. Typically, this field is used to distinguish minor revisions of a product.

This field has an IMPLEMENTATION DEFINED value.

Access to this field is **RO**.

Implementer, bits [11:0]

Contains the JEP106 code of the company that implemented the ITS:

- Bits [11:8] are the JEP106 continuation code of the implementer. For an Arm implementation, this field is 0x4.
- Bit [7] is always 0.
- Bits [6:0] are the JEP106 identity code of the implementer. For an Arm implementation, bits [7:0] are therefore 0x3B.

Accessing GITS_IIDR

GITS_IIDR can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
GIC ITS control	0x0004	GITS_IIDR

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

AccessesThis on interface thisis interfaceaccessible areas follows: **RO**.

30/09/2021 14:53:09; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

GITS_MPAMIDR, Report maximum PARTID and PMG Register

The GITS_MPAMIDR characteristics are:

Purpose

Reports the maximum support PARTID and PMG values.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GITS_MPAMIDR are RES0.

A copy of this register is provided for each ITS.

When [GITS_TYPER](#).MPAM==0, this register is RES0.

Attributes

GITS_MPAMIDR 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								PMGmax								PARTIDmax															

Bits [31:24]

Reserved, RES0.

PMGmax, bits [23:16]

Maximum PMG value supported.

PARTIDmax, bits [15:0]

Maximum PARTID value supported.

Accessing GITS_MPAMIDR

GITS_MPAMIDR can be accessed through the memory-mapped interfaces:

Component	Offset
GIC ITS control	0x0010

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RO**.

3020/09/2021 14:12:53.37: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

GITS_MPIDR, Report ITS's affinity.

The GITS_MPIDR characteristics are:

Purpose

Reports ITS's affinity when the vPE Table is shared with Redistributors.

Configuration

This register is present only when FEAT_GICv4p1 is implemented. Otherwise, direct accesses to GITS_MPIDR are RES0.

A copy of this register is provided for each ITS.

When [GITS_TYPER](#).SVPET==0, this register is RES0.

Attributes

GITS_MPIDR 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
Aff3								Aff2								Aff1								RES0							

Aff3, bits [31:24]

The Affinity level 3 value for the ITS.

Aff2, bits [23:16]

The Affinity level 2 value for the ITS.

Aff1, bits [15:8]

The Affinity level 1 value for the ITS.

Bits [7:0]

Reserved, RES0.

Accessing GITS_MPIDR

GITS_MPIDR can be accessed through the memory-mapped interfaces:

Component	Offset
GIC ITS control	0x0018

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RO**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

(old)

htmldiff from-

(new)

GITS_PARTIDR, Set PARTID and PMG Register

The GITS_PARTIDR characteristics are:

Purpose

Sets the PARTID and PMG values used for memory accesses by the ITS.

Configuration

This register is present only when FEAT_GICv3p1 is implemented. Otherwise, direct accesses to GITS_PARTIDR are RES0.

A copy of this register is provided for each ITS.

When [GITS_TYPER](#).MPAM==0, this register is RES0.

Attributes

GITS_PARTIDR 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								PARTID															

Bits [31:24]

Reserved, RES0.

PMG, bits [23:16]

PMG value used when ITS accesses memory.

It is IMPLEMENTATION DEFINED whether bits not needed to represent PMG values in the range 0 to PMG_MAX are stateful or RES0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

PARTID, bits [15:0]

PARTID value used when ITS accesses memory.

It is IMPLEMENTATION DEFINED whether bits not needed to represent PARTID values in the range 0 to PARTID_MAX are stateful or RES0.

The reset behavior of this field is:

- On a GIC reset, this field resets to 0.

Accessing GITS_PARTIDR

GITS_PARTIDR can be accessed through the memory-mapped interfaces:

Component	Offset
GIC ITS control	0x0014

- When GICD_CTLR.DS == 0 accesses to this register are **RW**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RW**.

3020/09/2021 1412:5237; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

- When GICD_CTLR.DS == 0 accesses to this register are **WO**.
- When an access is Secure accesses to this register are **WO**.
- When an access is Non-secure accesses to this register are **WO**.

Accesses This on interface this is interface accessible areas follows: **WO**.

3020/09/2021 14:12:5237: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

GITS_STATUSR, ITS Error Reporting Status Register

The GITS_STATUSR characteristics are:

Purpose

Provides software with a mechanism to detect:

- Accesses to reserved locations.
- Writes to read-only locations.
- Reads of write-only locations.
- Unmapped MSIs.

Configuration

Attributes

GITS_STATUSR 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																						Syndrome		Syndrome		Overflow	UMSI	WROD	RWOD	WRD	RRD

Bits [31:10]

Reserved, RES0.

SyndromeSyndrome, bits [9:6]

Syndrome for the MSI that set GITS_STATUSR.UMSI to 1.

SyndromeSyndrome	Meaning
0b0000	Unknown reason.
0b0010	DeviceID out of range.
0b0011	DeviceID unmapped.
0b0100	EventID out of range.
0b0101	EventID unmapped.
0b0111	Collection unmapped.
0b1001	vPEID unmapped.

An implementation might not support reporting all syndromes, and might report 0b0000 for any cause.

This field is UNKNOWN when GITS_STATUSR.UMSI is 0.

Overflow, bit [5]

Reports whether an unmapped MSI has been received while GITS_STATUSR.UMSI is 1.

Overflow	Meaning
0b0	No unmapped MSIs have been received since GITS_STATUSR.UMSI set to 1.
0b1	At least one unmapped MSIs have been received since GITS_STATUSR.UMSI set to 1.

A software write of 1 to the bit clears it. A write of any other value is ignored.

If [GITS_TYPER](#).UMSI is 0, this field is RES0.

UMSI, bit [4]

Reports whether an unmapped MSI has been received

An unmapped MSI is defined as an MSI arriving at [GITS_TRANSLATER](#) for which there is insufficient mapping information for it to be forwarded to a Redistributor.

It is IMPLEMENTATION DEFINED whether an INT command can be reported as an unmapped MSI.

UMSI	Meaning
0b0	No unmapped MSIs have been received.
0b1	Unmapped MSI received.

A software write of 1 to the bit clears it. A write of any other value is ignored.

If [GITS_TYPER](#).UMSI is 0, this field is RES0.

WROD, bit [3]

Write to an RO location.

WROD	Meaning
0b0	Normal operation.
0b1	A write to an RO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

RWOD, bit [2]

Read of a WO location.

RWOD	Meaning
0b0	Normal operation.
0b1	A read of a WO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

WRD, bit [1]

Write to a reserved location.

WRD	Meaning
0b0	Normal operation.
0b1	A write to a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

RRD, bit [0]

Read of a reserved location.

RRD	Meaning
0b0	Normal operation.
0b1	A read of a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

Accessing GITS_STATUSR

This is an optional register. If the register is not implemented, the location is RAZ/WI.

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fe44a233fbbdfb36e47856c443a7ce9a85f5e501ca

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

GITS_TRANSLATER, ITS Translation Register

The GITS_TRANSLATER characteristics are:

Purpose

Written by a requesting Device to signal an interrupt for translation by the ITS.

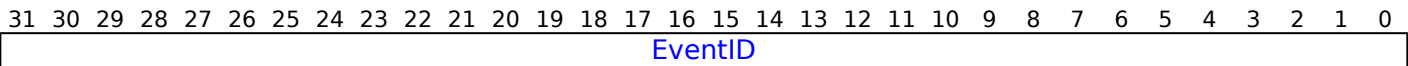
Configuration

This register is at the same offset as [GICD_SETSPI_NSR](#) in the Distributor, and is at the same offset as [GICR_SETLPIR](#) in the Redistributor.

Attributes

GITS_TRANSLATER is a 32-bit register.

Field descriptions



EventID, bits [31:0]

An identifier corresponding to the interrupt to be translated.

Note

The size of the EventID is DeviceID specific, and set when the DeviceID is mapped to an ITT (using MAPD).

The number of EventID bits implemented is reported by [GITS_TYPER.ID_bits](#). If a write specifies non-zero identifiers bits outside this range behavior is a CONSTRAINED UNPREDICTABLE choice between:

- Non-zero identifier bits outside the supported range are ignored.
- The write is ignored.

The DeviceID presented to an ITS is used to index a device table. The device table maps the DeviceID to an interrupt translation table for that device.

Accessing GITS_TRANSLATER

16-bit access to bits [15:0] of this register must be supported. When this register is written by a 16-bit transaction, bits [31:16] are written as zero.

Implementations must ensure that:

- A unique DeviceID is provided for each requesting device, and the DeviceID is presented to the ITS when a write to this register occurs in a manner that cannot be spoofed by any agent capable of performing writes.
- The DeviceID presented corresponds to the DeviceID field in the ITS commands.

Writes to this register are ignored if any of the following are true:

- [GITS_CTLR.Enabled](#) == 0.
- The presented DeviceID is not mapped to an Interrupt Translation Table.
- The DeviceID is larger than the supported size.

- The DeviceID is mapped to an Interrupt Translation Table, but the EventID is outside the range specified by MAPD.
- The EventID is mapped to an Interrupt Translation Table and the EventID is within the range specified by MAPD, but the EventID is unmapped.

Translation requests that result from writes to this register are subject to certain ordering rules. For more information, see 'Ordering of translations with the output to ITS commands' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

GITS_TRANSLATER can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
GIC ITS translation	0x0040	GITS_TRANSLATER

- When GICD_CTLR.DS == 0 accesses to this register are **WO**.
- When an access is Secure accesses to this register are **WO**.
- When an access is Non-secure accesses to this register are **WO**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **WO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The GITS_TYPER characteristics are:

Purpose

Specifies the features that an ITS supports.

Configuration

Attributes

GITS_TYPER 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					3																																	
RES0															INV		UMSI		irq		UMSI		inID		SVPET		VMAPP		VSGI		IMPAM		VMOVP		CIL		CIDb																																																	
HCC										RES0					PTASEIS		Devbits					ID_bits					ITT_entry_size					IMPLEMENTATION DEFINED					C																																																	
31			30			29			28			27			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 [63:47]

Reserved, RES0.

INV, bit [46]

ITS cache invalidation behavior on disable.

INV	Meaning
0b0	It is IMPLEMENTATION DEFINED whether ITS caches are invalidated on clearing GITS_CTLR.Enabled and GITS_BASER<n>.Valid .
0b1	ITS caches are invalidated on clearing GITS_CTLR.Enabled and GITS_BASER<n>.Valid .

If GITS_TYPER.INV is 1, after the following sequence:

- GITS_CTLR.Enabled written to 0.
- A read of GITS_CTLR.Quiescent returns 1.
- GITS_BASER<n>.Valid written to 0.

There is no cached information from the ITS memory structure pointed to by [GITS_BASER<n>](#).

UMSlirq, bit [45]

Indicates support for generating an interrupt on receiving unmapped MSI.

UMSIirq	Meaning
0b0	Interrupt on unmapped MSI not supported.
0b1	Interrupt on unmapped MSI is supported.

If GITS_TYPER.UMSI is 0, this field is RES0.

UMSI, bit [44]

Indicates support for reporting receipt of unmapped MSIs.

UMSI	Meaning
0b0	Reporting of unmapped MSIs is not supported.
0b1	Reporting of unmapped MSIs is supported.

nID, bit [43]

When FEAT_GICv4p1 is implemented:

nID

nID	Meaning
0b0	Individual doorbell interrupt supported.
0b1	Individual doorbell interrupt not supported.

Otherwise:

Reserved, RES0.

SVPET, bits [42:41]

When FEAT_GICv4p1 is implemented:

SVPET

SVPET	Meaning
0b00	vPE Table is not shared with Redistributors.
0b01	vPE Table is shared with the groups of Redistributors indicated by GITS_MPIDR.Aff3.
0b10	vPE Table is shared with the groups of Redistributors indicated by GITS_MPIDR fields Aff3 and Aff2.
0b11	vPE Table is shared with the groups of Redistributors indicated by GITS_MPIDR fields Aff3, Aff2 and Aff1.

Otherwise:

Reserved, RES0.

VMAPP, bit [40]

When FEAT_GICv4p1 is implemented:

VMAPP

VMAPP	Meaning
0b0	FEAT_GICv4 VMAPP command layout.
0b1	FEAT_GICv4p1 VMAPP command layout.

Otherwise:

Reserved, RES0.

VSGI, bit [39]

When FEAT_GICv4p1 is implemented:

VSGI

VSGI	Meaning
0b0	Direct injection of SGIs is not supported.
0b1	Direct injection of SGIs is supported.

Otherwise:

Reserved, RES0.

MPAM, bit [38]

When FEAT_GICv3p1 is implemented:

MPAM

MPAM	Meaning
0b0	MPAM is not supported.
0b1	MPAM is supported.

Otherwise:

Reserved, RES0.

VMOVP, bit [37]

Indicates the form of the VMOVP command.

VMOVP	Meaning
0b0	When moving a vPE, software must issue a VMOVP on all ITSs that have mappings for that vPE. The ITSList and Sequence Number fields in the VMOVP command must ensure synchronization, otherwise behavior is UNPREDICTABLE.
0b1	When moving a vPE, software must only issue a VMOVP on one of the ITSs that has a mapping for that vPE. The ITSList and Sequence Number fields in the VMOVP command are RES0.

CIL, bit [36]

Collection ID Limit.

CIL	Meaning
0b0	ITS supports 16-bit Collection ID, GITS_TYPER.CIDbits is RES0.
0b1	GITS_TYPER.CIDbits indicates supported Collection ID size

In implementations that do not support Collections in external memory, this bit is RES0 and the number of Collections supported is reported by [GITS_TYPER.HCC](#).

CIDbits, bits [35:32]

Number of Collection ID bits.

- The number of bits of Collection ID minus one.
- When [GITS_TYPER.CIL](#) == 0, this field is RES0.

HCC, bits [31:24]

Hardware Collection Count. The number of interrupt collections supported by the ITS without provisioning of external memory.

Note

Collections held in hardware are unmapped at reset.

Bits [23:20]

Reserved, RES0.

PTA, bit [19]

Physical Target Addresses. Indicates the format of the target address:

PTA	Meaning
0b0	The target address corresponds to the PE number specified by GICR_TYPER.Processor_Number .
0b1	The target address corresponds to the base physical address of the required Redistributor.

For more information, see 'RDbase' in ARM® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

SEIS, bit [18]

SEI support. Indicates whether the virtual CPU interface supports generation of SEIs:

SEIS	Meaning
0b0	The ITS does not support local generation of SEIs.
0b1	The ITS supports local generation of SEIs.

Devbits, bits [17:13]

The number of DeviceID bits implemented, minus one.

ID_bits, bits [12:8]

The number of EventID bits implemented, minus one.

ITT_entry_size, bits [7:4]

Read-only. Indicates the number of bytes per translation table entry, minus one.

For more information about the ITS command 'MAPD', see MAPD.

IMPLEMENTATION DEFINED, bit [3]

IMPLEMENTATION DEFINED.

CCT, bit [2]

Cumulative Collection Tables.

CCT	Meaning
0b0	The total number of supported collections is determined by the number of collections held in memory only.
0b1	The total number of supported collections is determined by number of collections that are held in memory and the number indicated by GITS_TYPER.HCC.

If GITS_TYPER.HCC == 0, or if memory backed collections are not supported (all [GITS_BASER<n>.Type](#) != 100), this bit is RES0.

Virtual, bit [1]

When FEAT_GICv4 is implemented:

Indicates whether the ITS supports virtual LPIs and direct injection of virtual LPIs:

Virtual	Meaning
0b0	The ITS does not support virtual LPIs or direct injection of virtual LPIs.
0b1	The ITS supports virtual LPIs and direct injection of virtual LPIs.

Otherwise:

Reserved, RES0.

Physical, bit [0]

Indicates whether the ITS supports physical LPis:

Physical	Meaning
0b0	The ITS does not support physical LPIs.
0b1	The ITS supports physical LPIs.

This field is RES1, indicating that the ITS supports physical LPs.

Accessing GITS_TYPER

GITS_TYPER can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
GIC ITS control	0x0008	GITS_TYPER

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RO**.
- When an access is Non-secure accesses to this register are **RO**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RO.**

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The GITS_UMSIR characteristics are:

Purpose

Provides the DeviceID and EventID of the unmapped MSI that set [GITS_STATUSR](#).UMSI.

Configuration

This register is present only when GITS_TYPER.UMSI == 1. Otherwise, direct accesses to GITS_UMSIR are RES0.

Attributes

GITS_UMSIR is a 64-bit register.

Field descriptions

Diagram illustrating the memory layout for the event data structure. The top 32 bits (63 to 32) are labeled. The bottom 32 bits (31 to 0) are labeled. The top 32 bits are divided into two 16-bit fields: 'DeviceID' (bits 48-63) and 'EventID' (bits 32-47).

DeviceID, bits [63:32]

DeviceID of MSI that set [GITS_STATUSR](#).UMSI to 1.

If [GITS_STATUS.UMSI](#) is 0, this field is UNKNOWN.

EventID, bits [31:0]

EventID of MSI that set [GITS_STATUSR.UMSI](#) to 1.

If [GITS_STATUSR](#).UMSI is 0, this field is UNKNOWN.

Accessing GITS UMSIR

GITS_UMSIR can be accessed through the memory-mapped interfaces:

Component	Offset	Instance
GIC ITS control	0x0048	GITS_UMSIR

- When GICD_CTLR.DS == 0 accesses to this register are **RO**.
- When an access is Secure accesses to this register are **RW**.
- When an access is Non-secure accesses to this register are **RW**.

AccessesThis oninterface thisis interfaceaccessible areas follows: **RO**.

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

no old file

htmldiff from-

(new)

MPAMCFG_CASSOC, MPAM Cache Maximum Associativity Partition Configuration Register

The MPAMCFG_CASSOC characteristics are:

Purpose

The MPAMCFG_CASSOC is a 32-bit read/write register that controls the maximum fraction of the cache associativity that the PARTID selected by [MPAMCFG_PART_SEL](#) is permitted to allocate.

MPAMCFG_CASSOC_s controls the cache maximum associativity for the Secure PARTID selected by the Secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_CASSOC_ns controls the cache maximum associativity for the Non-secure PARTID selected by the Non-secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_CASSOC_rl controls the cache maximum associativity for the Realm PARTID selected by the Realm instance of [MPAMCFG_PART_SEL](#). MPAMCFG_CASSOC_rt controls the cache maximum associativity for the Root PARTID selected by the Root instance of [MPAMCFG_PART_SEL](#).

If [MPAMF_IDR.HAS_RIS](#) is 1, the control settings accessed are those of the resource instance currently selected by [MPAMCFG_PART_SEL.RIS](#) and the PARTID selected by [MPAMCFG_PART_SEL.PARTID_SEL](#).

Configuration

The power domain of MPAMCFG_CASSOC is IMPLEMENTATION DEFINED.

This register is present only when [MPAMF_IDR.HAS_CCAP_PART](#) == 1, ([FEAT_MPAMv0p1](#) is implemented or [FEAT_MPAMv1p1](#) is implemented) and [MPAMF_CCAP_IDR.HAS_CASSOC](#) == 1. Otherwise, direct accesses to MPAMCFG_CASSOC are RES0.

Attributes

MPAMCFG_CASSOC 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																CASSOC															

Bits [31:16]

Reserved, RES0.

CASSOC, bits [15:0]

Maximum cache associativity usage in fixed-point fraction format by the partition selected by [MPAMCFG_PART_SEL](#). The fraction represents the portion of the cache associativity that the PARTID is permitted to allocate. CASSOC controls the fraction of associativity in each associativity grouping of the cache. In a set associative cache, CASSOC applies to the fraction of the ways in each set.

The implemented width of the fixed-point fraction is given in [MPAMF_CCAP_IDR.CASSOC_WD](#). Unimplemented bits within the field are RAZ/WI. The implemented bits of the CASSOC field are always the most significant bits of the field.

The fixed-point fraction CASSOC is less than 1. The implied binary point is between bits 15 and 16. This representation has as the largest fraction of the cache that can be represented in an implementation with w implemented bits is 1.0 minus one half to the power w.

Accessing MPAMCFG_CASSOC

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:

- MPAMCFG_CASSOC_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_CASSOC_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_CASSOC_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_CASSOC_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_CASSOC_s, MPAMCFG_CASSOC_ns, MPAMCFG_CASSOC_rt, and MPAMCFG_CASSOC_rl must be separate registers:

- The Secure instance (MPAMCFG_CASSOC_s) accesses the cache maximum associativity partitioning used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_CASSOC_ns) accesses the cache maximum associativity partitioning used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_CASSOC_rt) accesses the cache maximum associativity partitioning used for Root PARTIDs.
- The Realm instance (MPAMCFG_CASSOC_rl) accesses the cache maximum associativity partitioning used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_CASSOC access the cache maximum associativity partitioning configuration settings for the cache resource instance selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When RIS is not implemented, loads and stores to MPAMCFG_CASSOC access the cache maximum associativity partitioning configuration settings for the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When PARTID narrowing is implemented, loads and stores to MPAMCFG_CASSOC access the cache maximum associativity partitioning configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_CASSOC access the cache maximum associativity partitioning configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_CASSOC can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0118	MPAMCFG_CASSOC_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0118	MPAMCFG_CASSOC_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0118	MPAMCFG_CASSOC_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0118	MPAMCFG_CASSOC_rl

When FEAT_RME is implemented access on this interface are **RW**.

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

(old)

htmldiff from-

(new)

MPAMCFG_CMAX, MPAM Cache Maximum Capacity Partition Configuration Register

The MPAMCFG_CMAX characteristics are:

Purpose

The MPAMCFG_CMAX is a 32-bit read/write register that controls the maximum fraction of the cache capacity that the PARTID selected by [MPAMCFG_PART_SEL](#) is permitted to allocate.

MPAMCFG_CMAX_s controls the cache maximum capacity for the Secure PARTID selected by the Secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_CMAX_ns controls the cache maximum capacity for the Non-secure PARTID selected by the Non-secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_CMAX_rt controls the cache maximum capacity for the Root PARTID selected by the Root instance of [MPAMCFG_PART_SEL](#). MPAMCFG_CMAX_rl controls the cache maximum capacity for the Realm PARTID selected by the Realm instance of [MPAMCFG_PART_SEL](#).

If [MPAMF_IDR.HAS_RIS](#) is 1, the control settings accessed are those of the resource instance currently selected by [MPAMCFG_PART_SEL.RIS](#) and the PARTID selected by [MPAMCFG_PART_SEL.PARTID_SEL](#).

Configuration

The power domain of MPAMCFG_CMAX is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented, [MPAMF_IDR.HAS_CCAP_PART == 1](#) and [MPAMF_CCAP_IDR.NO_CMAX](#) [MPAMF_IDR.HAS_CCAP_PART == 0.1](#). Otherwise, direct accesses to MPAMCFG_CMAX are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMCFG_CMAX 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
SOFTLIMRES0																RES0CMAX															

SOFTLIM, Bits bit [31:16]

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_CCAP_IDR.HAS_CMAX_SOFTLIM == 1:

Soft limiting of CMAX. Soft limiting allows some allocations by a PARTID when its cache use is above the CMAX maximum cache capacity.

SOFTLIM	Meaning
0b0	When CMAX cache capacity is exceeded, the partition is not allowed to increase its cache capacity usage. It is only permitted to replace a line that was previously occupied by a line allocated by that PARTID.
0b1	When CMAX cache capacity is exceeded, the partition is permitted to allocate capacity beyond CMAX, but only from invalid lines or lines belonging to disabled PARTIDs.

Otherwise:

Reserved, RES0.

Bits [30:16]

Reserved, RES0.

CMAX, bits [15:0]

Maximum cache capacity usage in fixed-point fraction format by the partition selected by [MPAMCFG_PART_SEL](#). The fraction represents the portion of the total cache capacity that the PARTID is permitted to allocate.

The implemented width of the fixed-point fraction is given in [MPAMF_CCAP_IDR](#).CMAX_WD. Unimplemented bits within the field are RAZ/WI. The implemented bits of the CMAX field are always the most significant bits of the field.

The fixed-point fraction CMAX is less than 1. The implied binary point is between bits 15 and 16. This representation has as the largest fraction of the cache that can be represented in an implementation with w implemented bits is 1.0 minus one half to the power w.

Accessing MPAMCFG_CMAX

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:maps.~~

- MPAMCFG_CMAX_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_CMAX_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_CMAX_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_CMAX_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_CMAX_s, MPAMCFG_CMAX_ns, MPAMCFG_CMAX_rt, and MPAMCFG_CMAX_rl must be separate ~~registers:registers.~~

- The Secure instance (MPAMCFG_CMAX_s) accesses the cache capacity partitioning used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_CMAX_ns) accesses the cache capacity partitioning used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_CMAX_rt) accesses the cache capacity partitioning used for Root PARTIDs.
- The Realm instance (MPAMCFG_CMAX_rl) accesses the cache capacity partitioning used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_CMAX access the cache maximum capacity partitioning configuration settings for the cache resource instance selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When RIS is not implemented, loads and stores to MPAMCFG_CMAX access the cache maximum capacity partitioning configuration settings for the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When PARTID narrowing is implemented, loads and stores to MPAMCFG_CMAX access the cache maximum capacity partitioning configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_CMAX access the cache maximum capacity partitioning configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_CMAX can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0108	MPAMCFG_CMAX_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0108	MPAMCFG_CMAX_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0108	MPAMCFG_CMAX_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0108	MPAMCFG_CMAX_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

no old file

htmldiff from-

(new)

MPAMCFG_CMIN, MPAM Cache Minimum Capacity Partition Configuration Register

The MPAMCFG_CMIN characteristics are:

Purpose

The MPAMCFG_CMIN is a 32-bit read/write register that controls the fraction of the cache capacity that the PARTID selected by [MPAMCFG_PART_SEL](#) has priority to allocate.

MPAMCFG_CMIN_s controls the cache minimum capacity for the Secure PARTID selected by the Secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_CMIN_ns controls the cache minimum capacity for the Non-secure PARTID selected by the Non-secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_CMIN_rl controls the cache minimum capacity for the Realm PARTID selected by the Realm instance of [MPAMCFG_PART_SEL](#). MPAMCFG_CMIN_rt controls the cache minimum capacity for the Root PARTID selected by the Root instance of [MPAMCFG_PART_SEL](#).

If [MPAMF_IDR.HAS_RIS](#) is 1, the control settings accessed are those of the resource instance currently selected by [MPAMCFG_PART_SEL.RIS](#) and the PARTID selected by [MPAMCFG_PART_SEL.PARTID_SEL](#).

Configuration

The power domain of MPAMCFG_CMIN is IMPLEMENTATION DEFINED.

This register is present only when [MPAMF_IDR.HAS_CCAP_PART](#) == 1, (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and [MPAMF_CCAP_IDR.HAS_CMIN](#) == 1. Otherwise, direct accesses to MPAMCFG_CMIN are RES0.

Attributes

MPAMCFG_CMIN 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																CMIN															

Bits [31:16]

Reserved, RES0.

CMIN, bits [15:0]

Minimum cache capacity usage in fixed-point fraction format by the partition selected by [MPAMCFG_PART_SEL](#). The fraction represents the portion of the total cache capacity that the PARTID has priority to allocate.

The implemented width of the fixed-point fraction is the same as the width of [MPAMCFG_CMAX](#).CMAX which is given in [MPAMF_CCAP_IDR.CMAX_WD](#). Unimplemented bits within the field are RAZ/WI. The implemented bits of the CMIN field are always the most significant bits of the field.

The fixed-point fraction CMIN is less than 1. The implied binary point is between bits 15 and 16. This representation has as the largest fraction of the cache that can be represented in an implementation with w implemented bits is 1.0 minus one half to the power w.

Accessing MPAMCFG_CMIN

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:

- MPAMCFG_CMIN_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_CMIN_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_CMIN_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_CMIN_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_CMIN_s, MPAMCFG_CMIN_ns, MPAMCFG_CMIN_rt, and MPAMCFG_CMIN_rl must be separate registers:

- The Secure instance (MPAMCFG_CMIN_s) accesses the cache minimum capacity partitioning used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_CMIN_ns) accesses the cache minimum capacity partitioning used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_CMIN_rt) accesses the cache minimum capacity partitioning used for Root PARTIDs.
- The Realm instance (MPAMCFG_CMIN_rl) accesses the cache minimum capacity partitioning used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_CMIN access the cache minimum capacity partitioning configuration settings for the cache resource instance selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When RIS is not implemented, loads and stores to MPAMCFG_CMIN access the cache minimum capacity partitioning configuration settings for the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When PARTID narrowing is implemented, loads and stores to MPAMCFG_CMIN access the cache minimum capacity partitioning configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_CMIN access the cache minimum capacity partitioning configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_CMIN can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0110	MPAMCFG_CMIN_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0110	MPAMCFG_CMIN_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0110	MPAMCFG_CMIN_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0110	MPAMCFG_CMIN_rl

When FEAT_RME is implemented access on this interface are **RW**.

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

(old)

htmldiff from-

(new)

MPAMCFG_CPBM<n>, MPAM Cache Portion Bitmap Partition Configuration Register, n = 0 - 1023

The MPAMCFG_CPBM<n> characteristics are:

Purpose

The MPAMCFG_CPBM<n> register array gives access to the cache portion bitmap. Each register in the array is a read/write register that configures the cache portions numbered from <n * 32> to <31 + (n * 32)> that a PARTID is allowed to allocate.

After setting [MPAMCFG_PART_SEL](#) with a PARTID, software writes to the MPAMCFG_CPBM<n> register to configure which cache portions the PARTID is allowed to allocate.

The MPAMCFG_CPBM<n> register that contains the bitmap bit corresponding to cache portion p has n equal to p[15:5]. The field, P<x>, of that MPAMCFG_CPBM<n> register that contains the bitmap bit corresponding to cache portion p has x equal to p[4:0].

MPAMCFG_CPBM<n>_s controls cache portions for the Secure PARTID selected by the Secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_CPBM<n>_ns controls the cache portions for the Non-secure PARTID selected by the Non-secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_CPBM<n>_rt controls cache portions for the Root PARTID selected by the Root instance of [MPAMCFG_PART_SEL](#). MPAMCFG_CPBM<n>_rl controls the cache portions for the Realm PARTID selected by the Non-secure instance of [MPAMCFG_PART_SEL](#).

If [MPAMF_IDR.HAS_RIS](#) is 1, the control settings accessed are those of the resource instance currently selected by [MPAMCFG_PART_SEL.RIS](#) and the PARTID selected by [MPAMCFG_PART_SEL.PARTID_SEL](#).

Configuration

The power domain of MPAMCFG_CPBM<n> is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented and MPAMF_IDR.HAS_CPOR_PART == 1. Otherwise, direct accesses to MPAMCFG_CPBM<n> are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMCFG_CPBM<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25
P<32 * n + 31>	P<32 * n + 30>	P<32 * n + 29>	P<32 * n + 28>	P<32 * n + 27>	P<32 * n + 26>	P<32 * n + 25>

P<x + (n * 32)>, bit [x], for x = 31 to 0

Portion allocation control bit. Each cache portion allocation control bit, MPAMCFG_CPBM<n>.P<x>, grants permission to the PARTID selected by [MPAMCFG_PART_SEL](#) to allocate cache lines within cache portion <x + (n * 32)>.

P<x + (n * 32)>	Meaning
0b0	The PARTID is not permitted to allocate into cache portion <x + (n * 32)>.
0b1	The PARTID is permitted to allocate within cache portion <x + (n * 32)>.

The number of bits in the cache portion partitioning bit map of this component is given in [MPAMF_CPOR_IDR.CPBM_WD](#). CPBM_WD contains a value from 1 to 2^{15} , inclusive. Values of CPBM_WD greater than 32 require an array of 32-bit [MPAMCFG_CPBM<n>](#) registers to access the cache portion bitmap, up to 1024 registers.

Bits MPAMCFG_CPBM<n>.P<x + (n * 32)>, where <x + (n * 32)> is greater than or equal to CPBM_WD, are RES0:

- If $n > \text{MPAMF_CPOR_IDR.CPBM_WD}[15:5]$, the entire 32 P<x> are RES0.
- If $n == \text{MPAMF_CPOR_IDR.CPBM_WD}[15:5]$, bits [31: CPBM_WD[4:0]] are RES0 and the remaining bits are valid.
- If $n < \text{MPAMF_CPOR_IDR.CPBM_WD}[15:5]$, the entire 32 P<x> are valid.

Accessing MPAMCFG_CPBM<n>

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:maps](#).

- MPAMCFG_CPBM<n>_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_CPBM<n>_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_CPBM<n>_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_CPBM<n>_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_CPBM<n>_s, MPAMCFG_CPBM<n>_ns, MPAMCFG_CPBM<n>_rt, and MPAMCFG_CPBM<n>_rl must be separate [registers:registers](#).

- The Secure instance (MPAMCFG_CPBM<n>_s) accesses the cache portion bitmap used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_CPBM<n>_ns) accesses the cache portion bitmap used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_CPBM<n>_rt) accesses the cache portion bitmap used for Root PARTIDs.
- The Realm instance (MPAMCFG_CPBM<n>_rl) accesses the cache portion bitmap used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_CPBM<n> access the cache portion bitmap configuration settings for the cache resource instance selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When RIS is not implemented, loads and stores to MPAMCFG_CPBM<n> access the cache portion bitmap configuration settings for the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When PARTID narrowing is implemented, loads and stores to MPAMCFG_CPBM<n> access the cache portion bitmap configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_CPBM<n> access the cache portion bitmap configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_CPBM<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x1000 + (4 * n)	MPAMCFG_CPBM<n>_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x1000 + (4 * n)	MPAMCFG_CPBM<n>_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x1000 + (4 * n)	MPAMCFG_CPBM<n>_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x1000 + (4 * n)	MPAMCFG_CPBM<n>_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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	RES0															PARTID															

NFU, bit [31]

No Future Use.

NFU	Meaning
0b0	Control settings of the disabled PARTID must be retained.
0b1	Control settings of the disabled PARTID may take an UNKNOWN value.

Bits [30:16]

Reserved, RES0.

PARTID, bits [15:0]

Selects the PARTID to disable.

Accessing MPAMCFG_DIS

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:

- MPAMCFG_DIS_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_DIS_ns must be accessible from the Non-secure MPAM feature page.

- MPAMCFG_DIS_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_DIS_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_DIS_s, MPAMCFG_DIS_ns, MPAMCFG_DIS_rt, and MPAMCFG_DIS_rl must be separate registers:

- The Secure instance (MPAMCFG_DIS_s) accesses the PARTID disable used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_DIS_ns) accesses the PARTID disable used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_DIS_rt) accesses the PARTID disable used for Root PARTIDs.
- The Realm instance (MPAMCFG_DIS_rl) accesses the PARTID disable used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_DIS access the PARTID disable configuration settings for the PARTID disable resource instance selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When RIS is not implemented, loads and stores to MPAMCFG_DIS access the PARTID disable configuration settings for the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When PARTID narrowing is implemented, loads and stores to MPAMCFG_DIS access the PARTID disable configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_DIS access the PARTID disable configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_DIS can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0310	MPAMCFG_DIS_s

Accesses on this interface are **WO/RAZ**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0310	MPAMCFG_DIS_ns

Accesses on this interface are **WO/RAZ**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0310	MPAMCFG_DIS_rt

When FEAT_RME is implemented access on this interface are **WO/RAZ**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0310	MPAMCFG_DIS_rl

When FEAT_RME is implemented access on this interface are **WO/RAZ**.

30/09/2021 14:53; 092b4e1bbfbb45a293b198f9330c5f529ead2b0f

Copyright © 2010-2021 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)

MPAMCFG_EN, MPAM Partition Configuration Enable Register

The MPAMCFG_EN characteristics are:

Purpose

Enables a PARTID configuration as set in other MPAMCFG registers.

MPAMCFG_EN_s enables a Secure PARTID. MPAMCFG_EN_ns enables a Non-secure PARTID. MPAMCFG_EN_rl enables a Realm PARTID. MPAMCFG_EN_rt enables a Root PARTID.

Configuration

The power domain of MPAMCFG_EN 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_EN are RES0.

Attributes

MPAMCFG_EN 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																PARTID															

Bits [31:16]

Reserved, RES0.

PARTID, bits [15:0]

Selects the PARTID to enable.

Accessing MPAMCFG_EN

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:

- MPAMCFG_EN_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_EN_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_EN_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_EN_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_EN_s, MPAMCFG_EN_ns, MPAMCFG_EN_rt, and MPAMCFG_EN_rl must be separate registers:

- The Secure instance (MPAMCFG_EN_s) accesses the PARTID enable used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_EN_ns) accesses the PARTID enable used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_EN_rt) accesses the PARTID enable used for Root PARTIDs.
- The Realm instance (MPAMCFG_EN_rl) accesses the PARTID enable used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_EN access the PARTID enable configuration settings for the PARTID enable resource instance selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When RIS is not implemented, loads and stores to MPAMCFG_EN access the PARTID enable configuration settings for the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When PARTID narrowing is implemented, loads and stores to MPAMCFG_EN access the PARTID enable configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_EN access the PARTID enable configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_EN can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0300	MPAMCFG_EN_s

Accesses on this interface are **WO/RAZ**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0300	MPAMCFG_EN_ns

Accesses on this interface are **WO/RAZ**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0300	MPAMCFG_EN_rt

When FEAT_RME is implemented access on this interface are **WO/RAZ**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0300	MPAMCFG_EN_rl

When FEAT_RME is implemented access on this interface are **WO/RAZ**.

30/09/2021 14:53; 092b4e1bbfbb45a293b198f9330c5f529ead2b0f

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

MPAMCFG_EN_FLAGS, MPAM Partition Configuration Enable Flags Register

The MPAMCFG_EN_FLAGS characteristics are:

Purpose

Enable flags for 32 PARTIDs.

MPAMCFG_EN_FLAGS_s gives read/write access to 32 Secure PARTIDs. MPAMCFG_EN_FLAGS_ns gives read/write access to 32 Non-secure PARTIDs. MPAMCFG_EN_FLAGS_rl gives read/write access to 32 Realm PARTIDs. MPAMCFG_EN_FLAGS_rt gives read/write access to 32 Root PARTIDs.

Configuration

The power domain of MPAMCFG_EN_FLAGS 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_EN_FLAGS are RES0.

Attributes

MPAMCFG_EN_FLAGS 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
EN31	EN30	EN29	EN28	EN27	EN26	EN25	EN24	EN23	EN22	EN21	EN20	EN19	EN18	EN17	EN16	EN15	EN14	EN13	EN12	EN11

EN<x>, bit [x], for x = 31 to 0

PARTID Enable flags. The group of flags accessed is selected by [MPAMCFG_PART_SEL](#).PARTID & 0x0000001F in bit [0] to [MPAMCFG_PART_SEL](#).PARTID | 0x0000001F in bit [31].

EN<x>	Meaning
0b0	The PARTID is disabled.
0b1	The PARTID is enabled.

Each bit in [MPAMCFG_EN_FLAGS](#) gives access to the same state as controlled by [MPAMCFG_EN](#) and [MPAMCFG_DIS](#).

Bits MPAMCFG_EN_FLAGS.EN<x>, where ([MPAMCFG_PART_SEL](#).PARTID & 0x0000001F) + x is greater than [MPAMF_IDR](#).PARTID_MAX, are not required to be implemented.

As with other partitioning controls, the enable flag for PARTID 0 must be reset to 0b1 (enabled).

Accessing MPAMCFG_EN_FLAGS

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:

- MPAMCFG_EN_FLAGS_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_EN_FLAGS_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_EN_FLAGS_rt must be accessible from the Root MPAM feature page.

- MPAMCFG_EN_FLAGS_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_EN_FLAGS_s, MPAMCFG_EN_FLAGS_ns, MPAMCFG_EN_FLAGS_rt, and MPAMCFG_EN_FLAGS_rl must be separate registers:

- The Secure instance (MPAMCFG_EN_FLAGS_s) accesses the PARTID enable used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_EN_FLAGS_ns) accesses the PARTID enable used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_EN_FLAGS_rt) accesses the PARTID enable used for Root PARTIDs.
- The Realm instance (MPAMCFG_EN_FLAGS_rl) accesses the PARTID enable used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_EN_FLAGS access the PARTID enable configuration settings for the PARTID enable resource instance selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When RIS is not implemented, loads and stores to MPAMCFG_EN_FLAGS access the PARTID enable configuration settings for the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When PARTID narrowing is implemented, loads and stores to MPAMCFG_EN_FLAGS access the PARTID enable configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_EN_FLAGS access the PARTID enable configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_EN_FLAGS can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0320	MPAMCFG_EN_FLAGS_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0320	MPAMCFG_EN_FLAGS_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0320	MPAMCFG_EN_FLAGS_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0320	MPAMCFG_EN_FLAGS_rl

When FEAT_RME is implemented access on this interface are **RW**.

30/09/2021 14:53; 092b4e1bbfbb45a293b198f9330c5f529ead2b0f

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

no old file

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MPAMCFG_INTPARTID, MPAM Internal PARTID Narrowing Configuration Register

The MPAMCFG_INTPARTID characteristics are:

Purpose

MPAMCFG_INTPARTID is a 32-bit read/write register that controls the mapping of the PARTID selected by [MPAMCFG_PART_SEL](#) into a narrower internal PARTID (intPARTID).

MPAMCFG_INTPARTID_s controls the mapping for the Secure PARTID selected by the Secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_INTPARTID_ns controls the mapping for the Non-secure PARTID selected by the Non-secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_INTPARTID_rt controls the mapping for the Root PARTID selected by the Root instance of [MPAMCFG_PART_SEL](#). MPAMCFG_INTPARTID_rl controls the mapping for the Realm PARTID selected by the Realm instance of [MPAMCFG_PART_SEL](#).

The MPAMCFG_INTPARTID register associates the request PARTID (reqPARTID) in the [MPAMCFG_PART_SEL](#) register with an internal PARTID (intPARTID) in this register. To set that association, store reqPARTID into the [MPAMCFG_PART_SEL](#) register and then store the intPARTID into the MPAMCFG_INTPARTID register. To read the association, store reqPARTID into the MPAMCFG_PART_SEL register and then read MPAMCFG_INTPARTID.

If the intPARTID stored into MPAMCFG_INTPARTID is out-of-range or does not have the INTERNAL bit set, the association of reqPARTID to intPARTID is not written and [MPAMF_ESR](#) is set to indicate an intPARTID_Range error.

If [MPAMCFG_PART_SEL](#).INTERNAL is 1 when MPAMCFG_INTPARTID is read or written, [MPAMF_ESR](#) is set to indicate an Unexpected_INTERNAL error.

Configuration

The power domain of MPAMCFG_INTPARTID is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented and MPAMF_IDR.HAS_PARTID_NRW == 1. Otherwise, direct accesses to MPAMCFG_INTPARTID are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMCFG_INTPARTID 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																INTERNAL		INTPARTID													

Bits [31:17]

Reserved, RES0.

INTERNAL, bit [16]

Internal PARTID flag.

This bit must be 1 when written to the register. If written as 0, the write will not update the reqPARTID to intPARTID association.

On a read of this register, the bit will always read the value last written.

INTPARTID, bits [15:0]

This field contains the intPARTID mapped to the reqPARTID in [MPAMCFG_PART_SEL](#).

The maximum intPARTID supported is [MPAMF_PARTID_NRW_IDR](#).INTPARTID_MAX.

Accessing MPAMCFG_INTPARTID

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:maps](#).

- MPAMCFG_INTPARTID_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_INTPARTID_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_INTPARTID_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_INTPARTID_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_INTPARTID_s, MPAMCFG_INTPARTID_ns, MPAMCFG_INTPARTID_rt, and MPAMCFG_INTPARTID_rl must be separate [registers:registers](#).

- The Secure instance (MPAMCFG_INTPARTID_s) accesses the PARTID narrowing used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_INTPARTID_ns) accesses the PARTID narrowing used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_INTPARTID_rt) accesses the PARTID narrowing used for Root PARTIDs.
- The Realm instance (MPAMCFG_INTPARTID_rl) accesses the PARTID narrowing used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_INTPARTID access the PARTID narrowing configuration settings without being affected by [MPAMCFG_PART_SEL](#).RIS.

Loads and stores to MPAMCFG_INTPARTID access the PARTID narrowing configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_INTPARTID can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0600	MPAMCFG_INTPARTID_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0600	MPAMCFG_INTPARTID_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0600	MPAMCFG_INTPARTID_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0600	MPAMCFG_INTPARTID_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

Page 1584

MAX, bits [15:0]

Memory maximum bandwidth allocated to the partition selected by [MPAMCFG_PART_SEL](#). MAX is in fixed-point fraction format. The fraction represents the portion of the total memory bandwidth capacity through the controlled component that the PARTID is permitted to allocate.

The implemented width of the fixed-point fraction is given in [MPAMF_MBW_IDR.BWA_WD](#). Unimplemented bits are RAZ/WI. The implemented bits of the MAX field are always to the left of the field. For example, if BWA_WD = 3, the implemented bits are MPAMCFG_MBW_MAX[15:13] and MPAMCFG_MBW_MAX[12:0] are unimplemented.

The fixed-point fraction MAX is less than 1. The implied binary point is between bits 15 and 16. This representation has as the largest fraction of the bandwidth that can be represented in an implementation with w implemented bits is 1.0 minus one half to the power w.

Accessing MPAMCFG_MBW_MAX

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:maps](#).

- MPAMCFG_MBW_MAX_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_MBW_MAX_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_MBW_MAX_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_MBW_MAX_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_MBW_MAX_s, MPAMCFG_MBW_MAX_ns, MPAMCFG_MBW_MAX_rt, and MPAMCFG_MBW_MAX_rl must be separate [registers:registers](#).

- The Secure instance (MPAMCFG_MBW_MAX_s) accesses the memory maximum bandwidth partitioning used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_MBW_MAX_ns) accesses the memory maximum bandwidth partitioning used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_MBW_MAX_rt) accesses the memory maximum bandwidth partitioning used for Root PARTIDs.
- The Realm instance (MPAMCFG_MBW_MAX_rl) accesses the memory maximum bandwidth partitioning used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_MBW_MAX access the memory maximum bandwidth partitioning configuration settings for the bandwidth resource instance selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When RIS is not implemented, loads and stores to MPAMCFG_MBW_MAX access the memory maximum bandwidth partitioning configuration settings for the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When PARTID narrowing is implemented, loads and stores to MPAMCFG_MBW_MAX access the memory maximum bandwidth partitioning configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_MBW_MAX access the memory maximum bandwidth partitioning configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_MBW_MAX can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0208	MPAMCFG_MBW_MAX_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0208	MPAMCFG_MBW_MAX_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
-----------	-------	--------	----------

MPAM	MPAMF_BASE_rt	0x0208	MPAMCFG_MBW_MAX_rt
------	---------------	--------	--------------------

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x0208	MPAMCFG_MBW_MAX_r1

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

```
htmldiff from-
```

(new)

(old)

htmldiff from-

(new)

MPAMCFG_MBW_MIN, MPAM Memory Bandwidth Minimum Partition Configuration Register

The MPAMCFG_MBW_MIN characteristics are:

Purpose

MPAMCFG_MBW_MIN is a 32-bit read/write register that controls the minimum fraction of memory bandwidth that the PARTID selected by [MPAMCFG_PART_SEL](#) is permitted to use.

MPAMCFG_MBW_MIN_s controls the minimum bandwidth for the Secure PARTID selected by the Secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_MBW_MIN_ns controls the minimum bandwidth for the Non-secure PARTID selected by the Non-secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_MBW_MIN_rt controls the minimum bandwidth for the Root PARTID selected by the Root instance of [MPAMCFG_PART_SEL](#). MPAMCFG_MBW_MIN_rl controls the minimum bandwidth for the Realm PARTID selected by the Realm instance of [MPAMCFG_PART_SEL](#).

A PARTID that has used less than MIN is given preferential access to bandwidth.

If [MPAMF_IDR.HAS_RIS](#) is 1, the control settings accessed are those of the resource instance currently selected by [MPAMCFG_PART_SEL.RIS](#) and the PARTID selected by [MPAMCFG_PART_SEL.PARTID_SEL](#).

Configuration

The power domain of MPAMCFG_MBW_MIN is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented, MPAMF_IDR.HAS_MBW_PART == 1 and MPAMF_MBW_IDR.HAS_MIN == 1. Otherwise, direct accesses to MPAMCFG_MBW_MIN are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMCFG_MBW_MIN 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																MIN															

Bits [31:16]

Reserved, RES0.

MIN, bits [15:0]

Memory minimum bandwidth allocated to the partition selected by [MPAMCFG_PART_SEL](#). MIN is in fixed-point fraction format. The fraction represents the portion of the total memory bandwidth capacity through the controlled component that the PARTID is permitted to allocate.

The implemented width of the fixed-point fraction is given in [MPAMF_MBW_IDR.BWA_WD](#). Unimplemented bits are RAZ/WI. The implemented bits of the MIN field are always to the left of the field. For example, if BWA_WD = 4, the implemented bits are MPAMCFG_MBW_MIN[15:12] and MPAMCFG_MBW_MIN[11:0] are unimplemented.

The fixed-point fraction MIN is less than 1. The implied binary point is between bits 15 and 16. This representation has as the largest fraction of the bandwidth that can be represented in an implementation with w implemented bits is 1.0 minus one half to the power w.

Accessing MPAMCFG_MBW_MIN

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:maps.**

- MPAMCFG_MBW_MIN_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_MBW_MIN_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_MBW_MIN_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_MBW_MIN_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_MBW_MIN_s, MPAMCFG_MBW_MIN_ns, MPAMCFG_MBW_MIN_rt, and MPAMCFG_MBW_MIN_rl must be separate **registers:registers.**

- The Secure instance (MPAMCFG_MBW_MIN_s) accesses the memory minimum bandwidth partitioning used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_MBW_MIN_ns) accesses the memory minimum bandwidth partitioning used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_MBW_MIN_rt) accesses the memory minimum bandwidth partitioning used for Root PARTIDs.
- The Realm instance (MPAMCFG_MBW_MIN_rl) accesses the memory minimum bandwidth partitioning used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_MBW_MIN access the memory minimum bandwidth partitioning configuration settings for the bandwidth resource instance selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When RIS is not implemented, loads and stores to MPAMCFG_MBW_MIN access the memory minimum bandwidth partitioning configuration settings for the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When PARTID narrowing is implemented, loads and stores to MPAMCFG_MBW_MIN access the memory minimum bandwidth partitioning configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_MBW_MIN access the memory minimum bandwidth partitioning configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_MBW_MIN can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0200	MPAMCFG_MBW_MIN_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0200	MPAMCFG_MBW_MIN_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0200	MPAMCFG_MBW_MIN_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0200	MPAMCFG_MBW_MIN_rl

When FEAT_RME is implemented access on this interface are **RW**.

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

(old)

htmldiff from-

(new)

MPAMCFG_MBW_PBM<n>, MPAM Bandwidth Portion Bitmap Partition Configuration Register, n = 0 - 127

The MPAMCFG_MBW_PBM<n> characteristics are:

Purpose

The MPAMCFG_MBW_PBM<n> register array gives access to the memory bandwidth portion bitmap. Each register in the array is a read/write register that configures the bandwidth portions <32 * n> to <(32 * n) + 31> that a PARTID is allowed to allocate.

After setting [MPAMCFG_PART_SEL](#) with a PARTID, software writes to one or more of the MPAMCFG_MBW_PBM<n> registers to configure which bandwidth portions the PARTID is allowed to allocate.

The MPAMCFG_MBW_PBM<n> register that contains the bitmap bit corresponding to memory bandwidth portion p has n equal to $p[11:5]$. The field, $P<x + (32 * n)>$ of that MPAMCFG_MBW_PBM<n> register that contains the bitmap bit corresponding to memory bandwidth portion p has x equal to $p[4:0]$.

The MPAMCFG_MBW_PBM<n>_s registers control the bandwidth portion bitmap for the Secure PARTID selected by the Secure instance of [MPAMCFG_PART_SEL](#). The MPAMCFG_MBW_PBM<n>_ns registers control the bandwidth portion bitmap for the Non-secure PARTID selected by the Non-secure instance of [MPAMCFG_PART_SEL](#). The MPAMCFG_MBW_PBM<n>_rt registers control the bandwidth portion bitmap for the Root PARTID selected by the Root instance of [MPAMCFG_PART_SEL](#). The MPAMCFG_MBW_PBM<n>_rl registers control the bandwidth portion bitmap for the Realm PARTID selected by the Realm instance of [MPAMCFG_PART_SEL](#).

If [MPAMF_IDR](#).HAS_RIS is 1, the control settings accessed are those of the resource instance currently selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

Configuration

The power domain of MPAMCFG_MBW_PBM<n> is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented, MPAMF_IDR.HAS_MBW_PART == 1 and MPAMF_MBW_IDR.HAS_PBM == 1. Otherwise, direct accesses to MPAMCFG_MBW_PBM<n> are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMCFG_MBW_PBM<n> is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25
P<32 * n + 31>	P<32 * n + 30>	P<32 * n + 29>	P<32 * n + 28>	P<32 * n + 27>	P<32 * n + 26>	P<32 * n + 25>

P<x + (32 * n)>, bit [x], for x = 31 to 0

Portion allocation control bit. Each bandwidth portion allocation control bit MPAMCFG_MBW_PBM<n>.P<x + (32 * n)> grants permission to the PARTID selected by [MPAMCFG_PART_SEL](#) to allocate bandwidth within bandwidth portion <x + (32 * n)>.

P<x + (32 * n)>	Meaning
0b0	The PARTID is not permitted to allocate into bandwidth portion <x + (32 * n)>.
0b1	The PARTID is permitted to allocate within bandwidth portion <x + (32 * n)>.

The number of bits in the bandwidth portion partitioning bit map of this component is given in [MPAMF_MBW_IDR.BWPBM_WD](#). BWPBM_WD contains a value from 1 to 2^{12} , inclusive. Values of BWPBM_WD greater than 32 require a group of 32-bit registers to access the bandwidth portion bitmap, up to 128 32-bit registers.

Bits MPAMCFG_MBW_PBM<n>.P<<x + (32 * n)>>, where <x + (32 * n)> is greater than or equal to BWPBM_WD are RES0:

- If $n > \text{MPAMF_MBW_IDR.BWPBM_WD}[11:5]$, the entire 32 P<x> are RES0.
- If $n == \text{MPAMF_MBW_IDR.BWPBM_WD}[11:5]$, bits [31: BWPBM_WD[4:0]] are RES0 and the remaining bits are valid.
- If $n < \text{MPAMF_MBW_IDR.BWPBM_WD}[11:5]$, the entire 32 P<x> are valid.

Accessing MPAMCFG_MBW_PBM<n>

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:maps](#).

- MPAMCFG_MBW_PBM<n>_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_MBW_PBM<n>_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_MBW_PBM<n>_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_MBW_PBM<n>_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_MBW_PBM<n>_s, MPAMCFG_MBW_PBM<n>_ns, MPAMCFG_MBW_PBM<n>_rt, and MPAMCFG_MBW_PBM<n>_rl must be separate [registers:registers](#).

- The Secure instance (MPAMCFG_MBW_PBM<n>_s) accesses the memory bandwidth portion bitmap used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_MBW_PBM<n>_ns) accesses the memory bandwidth portion bitmap used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_MBW_PBM<n>_rt) accesses the memory bandwidth portion bitmap used for Root PARTIDs.
- The Realm instance (MPAMCFG_MBW_PBM<n>_rl) accesses the memory bandwidth portion bitmap used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_MBW_PBM<n> access the memory bandwidth portion bitmap configuration settings for the bandwidth resource instance selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When RIS is not implemented, loads and stores to MPAMCFG_MBW_PBM<n> access the memory bandwidth portion bitmap configuration settings for the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When PARTID narrowing is implemented, loads and stores to MPAMCFG_MBW_PBM<n> access the memory bandwidth portion bitmap configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_MBW_PBM<n> access the memory bandwidth portion bitmap configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_MBW_PBM<n> can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x2000 + (4 * n)	MPAMCFG_MBW_PBM<n>_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x2000 + (4 * n)	MPAMCFG_MBW_PBM<n>_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x2000 + (4 * n)	MPAMCFG_MBW_PBM<n>_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x2000 + (4 * n)	MPAMCFG_MBW_PBM<n>_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The MPAMCFG_MBW_PROP characteristics are:

Controls the proportional stride of memory bandwidth that the PARTID selected by `MPAMCFG_PART_SEL` uses.

MPAMCFG_MBW_PROP_s controls the bandwidth proportional stride for the Secure PARTID selected by the Secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_MBW_PROP_ns controls the bandwidth proportional stride for the Non-secure PARTID selected by the Non-secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_MBW_PROP_rt controls the bandwidth proportional stride for the Root PARTID selected by the Root instance of [MPAMCFG_PART_SEL](#). MPAMCFG_MBW_PROP_rl controls the bandwidth proportional stride for the Realm PARTID selected by the Realm instance of [MPAMCFG_PART_SEL](#).

Proportional stride is a relative cost of bandwidth requested by one PARTID in relation to the costs of the bandwidths requested by each other PARTID also competing to use the bandwidth.

If [MPAMF_IDR.HAS_RIS](#) is 1, the control settings accessed are those of the resource instance currently selected by [MPAMCFG_PART_SEL.RIS](#) and the PARTID selected by [MPAMCFG_PART_SEL.PARTID_SEL](#).

The power domain of MPAMCFG_MBW_PROP is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented, MPAMF_IDR.HAS_MBW_PART == 1 and MPAMF_MBW_IDR.HAS_PROP == 1. Otherwise, direct accesses to MPAMCFG_MBW_PROP are RES0.

The power and reset domain of each MSC component is specific to that component.

MPAMCFG_MBW_PROP is a 32-bit register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN		RES0														STRIDEM1															

EN, bit [31]

Enable proportional stride bandwidth partitioning.

EN	Meaning
0b0	The selected partition is not regulated by proportional stride bandwidth partitioning.
0b1	The selected partition has bandwidth usage regulated by proportional stride bandwidth partitioning as controlled by STRIDEM1.

Bits [30:16]

Reserved, RES0.

STRIDEM1, bits [15:0]

Memory bandwidth stride minus 1 allocated to the partition selected by [MPAMCFG_PART_SEL](#). STRIDEM1 represents the normalized cost of bandwidth consumption by the partition.

The proportional stride partitioning control parameter is an unsigned integer representing the normalized cost to a partition for consuming bandwidth. Larger values have a larger cost and correspond to a lesser allocation of bandwidth while smaller values indicate a lesser cost and therefore a higher allocation of bandwidth.

The implemented width of STRIDEM1 is given in MPAMF_MBW_IDR.BWA_WD.

Accessing MPAMCFG_MBW_PROP

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:~~[maps](#).

- MPAMCFG_MBW_PROP_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_MBW_PROP_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_MBW_PROP_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_MBW_PROP_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_MBW_PROP_s, MPAMCFG_MBW_PROP_ns, MPAMCFG_MBW_PROP_rt, and MPAMCFG_MBW_PROP_rl must be separate ~~registers:~~[registers](#).

- The Secure instance (MPAMCFG_MBW_PROP_s) accesses the memory proportional stride bandwidth partitioning used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_MBW_PROP_ns) accesses the memory proportional stride bandwidth partitioning used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_MBW_PROP_rt) accesses the memory proportional stride bandwidth partitioning used for Root PARTIDs.
- The Realm instance (MPAMCFG_MBW_PROP_rl) accesses the memory proportional stride bandwidth partitioning used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_MBW_PROP access the memory proportional stride bandwidth partitioning configuration settings for the bandwidth resource instance selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When RIS is not implemented, loads and stores to MPAMCFG_MBW_PROP access the memory proportional stride bandwidth partitioning configuration settings for the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When PARTID narrowing is implemented, loads and stores to MPAMCFG_MBW_PROP access the memory proportional stride bandwidth partitioning configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_MBW_PROP access the memory proportional stride bandwidth partitioning configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_MBW_PROP can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0500	MPAMCFG_MBW_PROP_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0500	MPAMCFG_MBW_PROP_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0500	MPAMCFG_MBW_PROP_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x0500	MPAMCFG_MBW_PROP_r1

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The MPAMCFG_MBW_WINWD characteristics are:

Purpose

MPAMCFG_MBW_WINWD is a 32-bit register that shows and sets the value of the window width for the PARTID in [MPAMCFG_PART_SEL](#).

MPAMCFG_MBW_WINWD_s reads and controls the bandwidth control window width for the Secure PARTID selected by the Secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_MBW_WINWD_ns reads and controls the bandwidth control window width for the Non-secure PARTID selected by the Non-secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_MBW_WINWD_rt reads and controls the bandwidth control window width for the Root PARTID selected by the Root instance of [MPAMCFG_PART_SEL](#). MPAMCFG_MBW_WINWD_rl reads and controls the bandwidth control window width for the Real PARTID selected by the Realm instance of [MPAMCFG_PART_SEL](#).

MPAMCFG_MBW_WINWD is read-only if [MPAMF_MBW_IDR](#).WINDWR == 0, and the window width is set by the hardware, even if variable.

MPAMCFG_MBW_WINWD is read/write if [MPAMF_MBW_IDR](#).WINDWR == 1, permitting configuration of the window width for each PARTID independently on hardware that supports this functionality.

If [MPAMF_IDR.HAS_RIS](#) is 1, the control settings accessed are those of the resource instance currently selected by [MPAMCFG_PART_SEL.RIS](#) and the PARTID selected by [MPAMCFG_PART_SEL.PARTID_SEL](#).

Configuration

The power domain of MPAMCFG_MBW_WINWD is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented and MPAMF_IDR.HAS_MBW_PART == 1. Otherwise, direct accesses to MPAMCFG MBW WINWD are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMCFG MBW WINWD 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								US INT																US FRAC							

Bits [31:24]

Reserved, RES0.

US INT, bits [23:8]

Window width, integer microseconds.

This field reads (and sets) the integer part of the window width in microseconds for the PARTID selected by `MPAMCFG PART SEL`.

US_FRAC, bits [7:0]

Window width, fractional microseconds.

This field reads (and sets) the fractional part of the window width in microseconds for the PARTID selected by [MPAMCFG_PART_SEL](#).

Accessing MPAMCFG_MBW_WINWD

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:maps:~~

- MPAMCFG_MBW_WINWD_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_MBW_WINWD_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_MBW_WINWD_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_MBW_WINWD_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_MBW_WINWD_s, MPAMCFG_MBW_WINWD_ns, MPAMCFG_MBW_WINWD_rt, and MPAMCFG_MBW_WINWD_rl must be separate ~~registers:registers:~~

- The Secure instance (MPAMCFG_MBW_WINWD_s) accesses the window width used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_MBW_WINWD_ns) accesses the window width used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_MBW_WINWD_rt) accesses the window width used for Root PARTIDs.
- The Realm instance (MPAMCFG_MBW_WINWD_rl) accesses the window width used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_MBW_WINWD access the window width configuration settings for the bandwidth resource instance selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When RIS is not implemented, loads and stores to MPAMCFG_MBW_WINWD access the window width configuration settings for the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

When PARTID narrowing is implemented, loads and stores to MPAMCFG_MBW_WINWD access the window width configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_MBW_WINWD access the window width configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL, and [MPAMCFG_PART_SEL](#).INTERNAL must be 0.

MPAMCFG_MBW_WINWD can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0220	MPAMCFG_MBW_WINWD_s

This interface is accessible as follows:

- When MPAMF_MBW_IDR.WINDWR == 0 accesses to this register are **RO**.
- When MPAMF_MBW_IDR.WINDWR == 1 accesses to this register are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0220	MPAMCFG_MBW_WINWD_ns

This interface is accessible as follows:

- When MPAMF_MBW_IDR.WINDWR == 0 accesses to this register are **RO**.
- When MPAMF_MBW_IDR.WINDWR == 1 accesses to this register are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0220	MPAMCFG_MBW_WINWD_rt

This interface is accessible as follows:

- When FEAT_RME is implemented and MPAMF_MBW_IDR.WINDWR == 0 accesses to this register are **RO**.
- When FEAT_RME is implemented and MPAMF_MBW_IDR.WINDWR == 1 accesses to this register are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x0220	MPAMCFG_MBW_WINWD_r1

This interface is accessible as follows:

- When FEAT_RME is implemented and MPAMF_MBW_IDR.WINDWR == 0 accesses to this register are **RO**.
- When FEAT_RME is implemented and MPAMF_MBW_IDR.WINDWR == 1 accesses to this register are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffdbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MPAMCFG_PRI, MPAM Priority Partition Configuration Register

The MPAMCFG_PRI characteristics are:

Purpose

Controls the internal and downstream priority of requests attributed to the PARTID selected by [MPAMCFG_PART_SEL](#). MPAMCFG_PRI_s controls the priorities for the Secure PARTID selected by the Secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_PRI_ns controls the priorities for the Non-secure PARTID selected by the Non-secure instance of [MPAMCFG_PART_SEL](#). MPAMCFG_PRI_rt controls the priorities for the Root PARTID selected by the Root instance of [MPAMCFG_PART_SEL](#). MPAMCFG_PRI_rl controls the priorities for the Realm PARTID selected by the Realm instance of [MPAMCFG_PART_SEL](#). If [MPAMF_IDR](#).HAS_RIS is 1, the control settings accessed are those of the resource instance currently selected by [MPAMCFG_PART_SEL](#).RIS and the PARTID selected by [MPAMCFG_PART_SEL](#).PARTID_SEL.

Configuration

The power domain of MPAMCFG_PRI is IMPLEMENTATION DEFINED. This register is present only when FEAT_MPAM is implemented and MPAMF_IDR.HAS_PRI_PART == 1. Otherwise, direct accesses to MPAMCFG_PRI are RES0. The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMCFG_PRI 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
DSPRI																INTPRI															

DSPRI, bits [31:16]

Downstream priority. If [MPAMF_PRI_IDR](#).HAS_DSPRI == 0, bits of this field are RES0 as this field is not used. If [MPAMF_PRI_IDR](#).HAS_DSPRI == 1, this field is a priority value applied to downstream communications from this MSC for transactions of the partition selected by [MPAMCFG_PART_SEL](#). The implemented width of this field is [MPAMF_PRI_IDR](#).DSPRI_WD bits. If the implemented width is less than the width of this field, the least significant bits are used. The encoding of priority is 0-as-lowest or 0-as-highest priority according to the value of [MPAMF_PRI_IDR](#).DSPRI_0_IS_LOW.

INTPRI, bits [15:0]

Internal priority. If [MPAMF_PRI_IDR](#).HAS_INTPRI == 0, bits of this field are RES0 as this field is not used.

If [MPAMF_PRI_IDR.HAS_INTPRI](#) == 1, this field is a priority value applied internally inside this MSC for transactions of the partition selected by [MPAMCFG_PART_SEL](#).

The implemented width of this field is [MPAMF_PRI_IDR.INTPRI_WD](#) bits. If the implemented width is less than the width of this field, the least significant bits are used.

The encoding of priority is 0-as-lowest or 0-as-highest priority according to the value of [MPAMF_PRI_IDR.INTPRI_0_IS_LOW](#).

Accessing MPAMCFG_PRI

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:maps](#).

- MPAMCFG_PRI_s must be accessible from the Secure MPAM feature page.
- MPAMCFG_PRI_ns must be accessible from the Non-secure MPAM feature page.
- MPAMCFG_PRI_rt must be accessible from the Root MPAM feature page.
- MPAMCFG_PRI_rl must be accessible from the Realm MPAM feature page.

MPAMCFG_PRI_s, MPAMCFG_PRI_ns, MPAMCFG_PRI_rt, and MPAMCFG_PRI_rl must be separate [registers:registers](#).

- The Secure instance (MPAMCFG_PRI_s) accesses the priority partitioning used for Secure PARTIDs.
- The Non-secure instance (MPAMCFG_PRI_ns) accesses the priority partitioning used for Non-secure PARTIDs.
- The Root instance (MPAMCFG_PRI_rt) accesses the priority partitioning used for Root PARTIDs.
- The Realm instance (MPAMCFG_PRI_rl) accesses the priority partitioning used for Realm PARTIDs.

When RIS is implemented, loads and stores to MPAMCFG_PRI access the priority partitioning configuration settings for the priority resource instance selected by [MPAMCFG_PART_SEL.RIS](#) and the PARTID selected by [MPAMCFG_PART_SEL.PARTID_SEL](#).

When RIS is not implemented, loads and stores to MPAMCFG_PRI access the priority partitioning configuration settings for the PARTID selected by [MPAMCFG_PART_SEL.PARTID_SEL](#).

When PARTID narrowing is implemented, loads and stores to MPAMCFG_PRI access the priority partitioning configuration settings for the internal PARTID selected by [MPAMCFG_PART_SEL.PARTID_SEL](#), and [MPAMCFG_PART_SEL.INTERNAL](#) must be 1.

When PARTID narrowing is not implemented, loads and stores to MPAMCFG_PRI access the priority partitioning configuration settings for the request PARTID selected by [MPAMCFG_PART_SEL.PARTID_SEL](#), and [MPAMCFG_PART_SEL.INTERNAL](#) must be 0.

MPAMCFG_PRI can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0400	MPAMCFG_PRI_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0400	MPAMCFG_PRI_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0400	MPAMCFG_PRI_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0400	MPAMCFG_PRI_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

The MPAMF_CCAP_IDR characteristics are:

Purpose

Indicates the number of fractional bits in `MPAMCFG_CMAX.CMAX`.

MPAMF_CCAP_IDR_s indicates the number of fractional bits in the Secure instance of [MPAMCFG_CMAX](#).
 MPAMF_CCAP_IDR_ns indicates the number of fractional bits in the Non-secure instance of [MPAMCFG_CMAX](#).
 MPAMF_CCAP_IDR_rt indicates the number of fractional bits in the Root cache capacity control settings register field, [MPAMCFG_CMAX](#).CMAX. MPAMF_CCAP_IDR_rl indicates the number of fractional bits in the Realm cache capacity control settings register field, [MPAMCFG_CMAX](#).CMAX.

When [MPAMF_IDR.HAS_RIS](#) is 1, some fields in this register give information for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#). The description of every field that is affected by [MPAMCFG_PART_SEL.RIS](#) has information within the field description.

Configuration

The power domain of MPAMF_CCAP_IDR is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented and MPAMF_IDR.HAS_CCAP_PART == 1. Otherwise, direct accesses to MPAMF_CCAP_IDR are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMF_CCAP_IDR is a 32-bit register.

Field descriptions



HAS CMAX SOFTLIM, Bits bit [31:6]

When FEAT MPAMv0p1 is implemented or FEAT MPAMv1p1 is implemented:

Has soft limiting selection field in MPAMCFG CMAX.

HAS_CMAX_SOFTLIM	Meaning
0b0	If MPAMCFG_CMAX is implemented, it has no SOFTLIM field and the maximum capacity is controlled with a hard limit.
0b1	If MPAMCFG_CMAX is implemented, that register has a SOFTLIMIT field to select between hard or soft limiting to the CMAX parameter.

If RIS is implemented, this field indicates selectable limiting for the cache maximum capacity control for the resource instance selected by **MPAMCFG PART SEL.RIS**.

Otherwise:

Reserved, RES0.

NO_CMAX, bit [30]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

Does not have CMAX partitioning.

NO_CMAX	Meaning
0b0	MPAMCFG_CMAX is implemented.
0b1	MPAMCFG_CMAX is not implemented.

If RIS is implemented, this field indicates the absence of a cache maximum capacity partitioning control for the resource instance selected by MPAMCFG_PART_SEL.RIS.

Otherwise:

Reserved, RES0.

HAS_CMIN, bit [29]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

Has cache minimum capacity partitioning.

HAS_CMIN	Meaning
0b0	MPAMCFG_CMIN is not implemented.
0b1	MPAMCFG_CMIN is implemented.

If RIS is implemented, this field indicates the presence of a cache minimum capacity partitioning control for the resource instance selected by MPAMCFG_PART_SEL.RIS.

Otherwise:

Reserved, RES0.

HAS_CASSOC, bit [28]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

Has cache maximum associativity partitioning.

HAS_CASSOC	Meaning
0b0	MPAMCFG_CASSOC is not implemented.
0b1	MPAMCFG_CASSOC is implemented.

If RIS is implemented, this field indicates the presence of a cache maximum associativity partitioning control for the resource instance selected by MPAMCFG_PART_SEL.RIS.

Otherwise:

Reserved, RES0.

Bits [27:13]

Reserved, RES0.

CASSOC_WD, bits [12:8]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

Number of fractional bits implemented in the cache associativity partitioning control, [MPAMCFG_CASSOC.CASSOC](#), of this MSC. See [MPAMCFG_CASSOC](#).

If RIS is implemented, this field indicates the number of fractional bits in the cache capacity partitioning control for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#).

Otherwise:

Reserved, RES0.

Bits [7:6]

Reserved, RES0.

CMAX_WD, bits [5:0]

Number of fractional bits implemented in the cache capacity partitioning control, [MPAMCFG_CMAX.CMAX](#), of this device. See [MPAMCFG_CMAX](#).

This field must contain a value from 1 to 16, inclusive.

If RIS is implemented, this field indicates the number of fractional bits in the cache capacity partitioning control for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#).

Accessing MPAMF_CCAP_IDR

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_CCAP_IDR is read-only.

MPAMF_CCAP_IDR must be readable from the Non-secure, Secure, Root, and Realm MPAM feature pages.

MPAMF_CCAP_IDR is permitted to have the same contents when read from the Secure, Non-secure, Root, and Realm MPAM feature pages unless the register contents are different for the different versions:

- MPAMF_CCAP_IDR_s is permitted to have either the same or different contents to MPAMF_CCAP_IDR_ns, MPAMF_CCAP_IDR_rt, or MPAMF_CCAP_IDR_rl.
- MPAMF_CCAP_IDR_ns is permitted to have either the same or different contents to MPAMF_CCAP_IDR_rt or MPAMF_CCAP_IDR_rl.
- MPAMF_CCAP_IDR_rt is permitted to have either the same or different contents to MPAMF_CCAP_IDR_rl.

There must be separate registers in the Secure (MPAMF_CCAP_IDR_s), Non-secure (MPAMF_CCAP_IDR_ns), Root (MPAMF_CCAP_IDR_rt), and Realm (MPAMF_CCAP_IDR_rl) MPAM feature pages.

When [MPAMF_IDR.HAS_RIS](#) is 1, MPAMF_CCAP_IDR shows the configuration of cache capacity partitioning for the cache resource instance selected by [MPAMCFG_PART_SEL.RIS](#). Fields that mention RIS in their field descriptions have values that track the implemented properties of the resource instance. Fields that do not mention RIS are constant across all resource instances.

MPAMF_CCAP_IDR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0038	MPAMF_CCAP_IDR_s

Accesses on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0038	MPAMF_CCAP_IDR_ns

Accesses on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0038	MPAMF_CCAP_IDR_rt

When FEAT_RME is implemented access on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x0038	MPAMF_CCAP_IDR_r1

When FEAT_RME is implemented access on this interface are **RO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

The MPAMF CSUMON IDR characteristics are:

Indicates the number of cache storage usage monitor instances and other properties of the CSU monitoring.

MPAMF_CSUMON_IDR_s indicates the number and properties of Secure cache storage usage monitoring.
 MPAMF_CSUMON_IDR_ns indicates the number and properties of Non-secure cache storage usage monitoring.
 MPAMF_CSUMON_IDR_rt indicates the number and properties of Root cache storage usage monitoring.
 MPAMF_CSUMON_IDR_rl indicates the number and properties of Realm cache storage usage monitoring.

If `MPAMF_IDR.HAS_RIS` is 1, fields that mention RIS must reflect the properties of the resource instance currently selected by `MPAMCFG_PART_SEL.RIS`. Fields that do not mention RIS are constant across all resource instances.

The power domain of MPAMF_CSUMON_IDR 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 MPAMF_CSUMON_IDR are RES0.

The power and reset domain of each MSC component is specific to that component.

MPAMF_CSUMON_IDR is a 32-bit register.

31	30	29	28	27	26	25
HAS_CAPTURE	CSU_RO	HAS_XCLRES0	RES0	HAS_OFSR	HAS_OFLOW_LNKGRES0	HAS_OFSRNUM_MON

HAS_CAPTURE, bit [31]

The implementation supports copying an [MSMON_CSU](#) to the corresponding [MSMON_CSU_CAPTURE](#) on a capture event.

HAS_CAPTURE	Meaning
0b0	MSMON_CSU_CAPTURE is not implemented and there is no support for capture events in the CSU monitor.
0b1	The MSMON_CSU_CAPTURE register is implemented and the CSU monitor supports the capture event behavior.

If RIS is implemented, this field indicates that CSU monitor capture is implemented for the resource instance selected by [MPAMCFG PART SEL](#).RIS.

CSU_RO, bit [30]

The implementation of [MSMON CSU](#) is read-only.

CSU_RO	Meaning
0b0	MSMON_CSU is read/write.
0b1	MSMON_CSU is read-only.

If RIS is implemented, this field indicates that the [MSMON_CSU](#) monitor register is read-only for the resource instance selected by [MPAMCFG_PART_SEL](#).RIS.

HAS_XCL, Bits bit [29:27]

When [FEAT_MPAMv0p1](#) is implemented or [FEAT_MPAMv1p1](#) is implemented:

Has filtering to exclude clean data and implements the [MSMON_CFG_CSU_FLT.XCL](#) field.

HAS_XCL	Meaning
0b0	MSMON_CFG_CSU_FLT does not implement the XCL field.
0b1	MSMON_CFG_CSU_FLT implements the XCL field to exclude counting data in the clean state in the monitor instance.

If RIS is implemented, this field indicates that the [MSMON_CFG_CSU_FLT.XCL](#) field is implemented in the CSU monitor instances for the resource instance selected by [MPAMCFG_PART_SEL](#).RIS.

Otherwise:

Reserved, RES0.

Bit [28]

Reserved, RES0.

HAS_OFLOW_LNKG, bit [27]

When [FEAT_MPAMv0p1](#) is implemented or [FEAT_MPAMv1p1](#) is implemented:

Supports [MSMON_CFG_CSU_CTL.OFLOW_LNKG](#) field to control how overflow on an instance affects other monitor instances in this MSC.

HAS_OFLOW_LNKG	Meaning
0b0	Does not support CSU overflow linkage.
0b1	Supports CSU overflow linkage and the MSMON_CFG_CSU_CTL.OFLOW_LNKG field.

If RIS is implemented, this field indicates that [MSMON_CFG_CSU_CTL.OFLOW_LNKG](#) is implemented for the resource instance selected by [MPAMCFG_PART_SEL](#).RIS.

Otherwise:

Reserved, RES0.

HAS_OFSR, bit [26]

When [FEAT_MPAMv0p1](#) is implemented or [FEAT_MPAMv1p1](#) is implemented:

The CSU monitor overflow status bitmap register, [MSMON_CSU_OFSR](#), is implemented.

HAS_OFSR	Meaning
0b0	MSMON_CSU_OFSR register is not implemented.
0b1	MSMON_CSU_OFSR register is implemented.

If RIS is implemented, this field indicates that CSU monitor overflow status bitmap register is implemented for the resource instance selected by [MPAMCFG_PART_SEL](#).RIS.

Otherwise:

Reserved, RES0.

HAS_OFLOW_CAPT, Bits bit [25:16]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

Supports [MSMON_CFG_CSU_CTL.OFLOW_CAPT](#) field to transfer the CSU monitor instance to its capture register on an overflow or overflow linkage event.

HAS_OFLOW_CAPT	Meaning
0b0	Does not support capture on overflow.
0b1	Supports capture on overflow and the MSMON_CFG_CSU_CTL.OFLOW_CAPT field.

If RIS is implemented, this field indicates that [MSMON_CFG_CSU_CTL.OFLOW_CAPT](#) is implemented for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#).

Otherwise:

Reserved, RES0.

Bits [24:16]

Reserved, RES0.

NUM_MON, bits [15:0]

The number of cache storage usage monitor instances implemented.

The largest [MSMON_CFG_MON_SEL.MON_SEL](#) value is NUM_MON minus 1.

If RIS is implemented, this field indicates the number of CSU monitor instances implemented for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#).

Accessing MPAMF_CSUMON_IDR

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_CSUMON_IDR is read-only.

MPAMF_CSUMON_IDR must be readable from the Non-secure, Secure, Root, and Realm MPAM feature pages.

MPAMF_CSUMON_IDR is permitted to have the same contents when read from the Secure, Non-secure, Root, and Realm MPAM feature pages unless the register contents are different for the different versions:

- MPAMF_CSUMON_IDR_s is permitted to have either the same or different contents to MPAMF_CSUMON_IDR_ns, MPAMF_CSUMON_IDR_rt, or MPAMF_CSUMON_IDR_rl.
- MPAMF_CSUMON_IDR_ns is permitted to have either the same or different contents to MPAMF_CSUMON_IDR_rt or MPAMF_CSUMON_IDR_rl.
- MPAMF_CSUMON_IDR_rt is permitted to have either the same or different contents to MPAMF_CSUMON_IDR_rl.

There must be separate registers in the Secure (MPAMF_CSUMON_IDR_s), Non-secure (MPAMF_CSUMON_IDR_ns), Root (MPAMF_CSUMON_IDR_rt), and Realm (MPAMF_CSUMON_IDR_rl) MPAM feature pages.

When [MPAMF_IDR.HAS_RIS](#) is 1, MPAMF_CSUMON_IDR shows the configuration of cache storage usage monitoring for the cache resource instance selected by [MPAMCFG_PART_SEL.RIS](#). Fields that mention RIS in their field descriptions have values that track the implemented properties of the resource instance. Fields that do not mention RIS are constant across all resource instances.

Access to MPAMF_CSUMON_IDR is not affected by [MSMON_CFG_MON_SEL.RIS](#).

Accesses on this interface are **RO**.

Accesses on this interface are **RO**.

When FEAT_RME is implemented access on this interface are **RO**.

When FEAT_RME is implemented access on this interface are **RO**.

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

Page 1609

The MPAMF_ECR characteristics are:

Purpose

MPAMF_ECR is a 32-bit read/write register that controls MPAM error interrupts for this MSC.

MPAMF_ECR_s controls Secure MPAM error handling. MPAMF_ECR_ns controls Non-secure MPAM error handling. MPAMF_ECR_rt controls Root MPAM error handling. MPAMF_ECR_rl controls Realm MPAM error handling.

Configuration

The power domain of MPAMF_ECR is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented. Otherwise, direct accesses to MPAMF_ECR are RES0.

If an MSC cannot encounter any of the error conditions listed in 'Errors in MSCs' in Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A (ARM DDI 0598), both the [MPAMF_ESR](#) and MPAMF_ECR must be RAZ/WI.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMF_ECR 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 INTEN

Bits [31:1]

Reserved, RES0.

INTEN, bit [0]

Interrupt Enable.

INTEN	Meaning
0b0	MPAM error interrupts are not signaled.
0b1	MPAM error interrupts are signaled.

Accessing MPAMF_ECR

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:~~ **maps:**

- MPAMF_ECR_s must be accessible from the Secure MPAM feature page.
- MPAMF_ECR_ns must be accessible from the Non-secure MPAM feature page.
- MPAMF_ECR_rt must be accessible from the Root MPAM feature page.
- MPAMF_ECR_rl must be accessible from the Realm MPAM feature page.

MPAMF_ECR_s, MPAMF_ECR_ns, MPAMF_ECR_rt, and MPAMF_ECR_rl must be separate ~~registers:registers.~~

- The Secure instance (MPAMF_ECR_s) accesses the error interrupt controls used for Secure PARTIDs.
- The Non-secure instance (MPAMF_ECR_ns) accesses the error interrupt controls used for Non-secure PARTIDs.
- The Root instance (MPAMF_ECR_rt) accesses the error interrupt controls used for Root PARTIDs.
- The Realm instance (MPAMF_ECR_rl) accesses the error interrupt controls used for Realm PARTIDs.

MPAMF_ECR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x00F0	MPAMF_ECR_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x00F0	MPAMF_ECR_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x00F0	MPAMF_ECR_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x00F0	MPAMF_ECR_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

- 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 on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x00E4	MPAMF_ERR_MSI_ADDR_H_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x00E4	MPAMF_ERR_MSI_ADDR_H_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x00E4	MPAMF_ERR_MSI_ADDR_H_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 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_L, MPAM Error MSI Low-part Address Register

The MPAMF_ERR_MSI_ADDR_L characteristics are:

Purpose

MPAMF_ERR_MSI_ADDR_L is a 32-bit read/write register for the low part of the MPAM error MSI address.

MPAMF_ERR_MSI_ADDR_L_s is the low part of the MSI write address for error interrupts related to Secure PARTIDs. MPAMF_ERR_MSI_ADDR_L_ns is the low part of the MSI write address for error interrupts related to Non-secure PARTIDs. MPAMF_ERR_MSI_ADDR_L_rt is the low part of the MSI write address for error interrupts related to Root PARTIDs. MPAMF_ERR_MSI_ADDR_L_rl is the low part of the MSI write address for error interrupts related to Realm PARTIDs.

Configuration

The power domain of MPAMF_ERR_MSI_ADDR_L is IMPLEMENTATION DEFINED.

This register is present only when MPAMF_IDR.HAS_ERR_MSI == 1. Otherwise, direct accesses to MPAMF_ERR_MSI_ADDR_L are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMF_ERR_MSI_ADDR_L 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 ADDR L																Bits[1:0]															

MSI_ADDR_L, bits [31:2]

MSI write address bits[31:2].

Bits [1:0]

Reads as 0b00.

Access to this field is **RO**.

Accessing MPAMF_ERR_MSI_ADDR_L

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:maps**.

- MPAMF_ERR_MSI_ADDR_L_s must be accessible from the Secure MPAM feature page.
- MPAMF_ERR_MSI_ADDR_L_ns must be accessible from the Non-secure MPAM feature page.
- MPAMF_ERR_MSI_ADDR_L_rt must be accessible from the Root MPAM feature page.
- MPAMF_ERR_MSI_ADDR_L_rl must be accessible from the Realm MPAM feature page.

MPAMF_ERR_MSI_ADDR_L_s, MPAMF_ERR_MSI_ADDR_L_ns, MPAMF_ERR_MSI_ADDR_L_rt, and MPAMF_ERR_MSI_ADDR_L_rl must be separate registers: registers.

- 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.

MPAMF_ERR_MSI_ADDR_L can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF BASE s	0x00E0	MPAMF ERR MSI ADDR L s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE ns	0x00E0	MPAMF_ERR_MSI_ADDR_L ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE rt	0x00E0	MPAMF_ERR_MSI_ADDR_L rt

When FEAT RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF BASE r1	0x00E0	MPAMF ERR MSI ADDR L r1

When FEAT RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MPAMF_ERR_MSI_ATTR, MPAM Error MSI Write Attributes Register

The MPAMF_ERR_MSI_ATTR characteristics are:

Purpose

MPAMF_ERR_MSI_ATTR is a 32-bit read/write register that controls MPAM error MSI write attributes for MPAM errors in this MSC.

MPAMF_ERR_MSI_ATTR_s controls the attributes of Secure MPAM error MSI writes. MPAMF_ERR_MSI_ATTR_ns controls the attributes of Non-secure MPAM error MSI writes. MPAMF_ERR_MSI_ATTR_rt controls the attributes of Root MPAM error MSI writes. MPAMF_ERR_MSI_ATTR_rl controls the attributes of Realm MPAM error MSI writes.

Configuration

The power domain of MPAMF_ERR_MSI_ATTR is IMPLEMENTATION DEFINED.

This register is present only when MPAMF_IDR.HAS_ERR_MSI == 1. Otherwise, direct accesses to MPAMF_ERR_MSI_ATTR are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMF_ERR_MSI_ATTR 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_SH	MSI_MEMATTR	RES0																												MSIEN

Bits [31:30]

Reserved, RES0.

MSI_SH, bits [29:28]

Sharability attribute of MSI writes.

MSI_SH	Meaning
0b00	Non-shareable.
0b01	Reserved, CONSTRAINED UNPREDICTABLE.
0b10	Outer Shareable.
0b11	Inner Shareable.

When MPAMF_ERR_MSI_ATTR.MSI_MEMATTR specifies a Device memory type, the contents of this field are IGNORED and Shareability is effectively Outer Shareable.

MSI_MEMATTR, bits [27:24]

Memory attributes of MSI writes.

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:maps.**

- MPAMF_ERR_MSI_ATTR_s must be accessible from the Secure MPAM feature page.
- MPAMF_ERR_MSI_ATTR_ns must be accessible from the Non-secure MPAM feature page.
- MPAMF_ERR_MSI_ATTR_rt must be accessible from the Root MPAM feature page.
- MPAMF_ERR_MSI_ATTR_rl must 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: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 on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x00EC	MPAMF_ERR_MSI_ATTR_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x00EC	MPAMF_ERR_MSI_ATTR_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x00EC	MPAMF_ERR_MSI_ATTR_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ee9a85f5e501ca

Copyright © 2010-2021 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 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:maps.**

- MPAMF_ERR_MSI_DATA_s must be accessible from the Secure MPAM feature page.
- MPAMF_ERR_MSI_DATA_ns must be accessible from the Non-secure MPAM feature page.
- MPAMF_ERR_MSI_DATA_rt must be accessible from the Root MPAM feature page.
- MPAMF_ERR_MSI_DATA_rl must 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: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 on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x00E8	MPAMF_ERR_MSI_DATA_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x00E8	MPAMF_ERR_MSI_DATA_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x00E8	MPAMF_ERR_MSI_DATA_r1

When FEAT_RME is implemented access on this interface are **RW**.

30/09/2021 14:53:37; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MPAMF_ERR_MSI_MPAM, MPAM Error MSI Write MPAM Information Register

The MPAMF_ERR_MSI_MPAM characteristics are:

Purpose

MPAMF_ERR_MSI_MPAM is a 32-bit read/write register that sets the MPAM information for error MSI write attributes for MPAM errors in this MSC.

MPAMF_ERR_MSI_MPAM_s controls MPAM information labeling of Secure MPAM error MSI writes.
 MPAMF_ERR_MSI_MPAM_ns controls MPAM information labeling of Non-secure MPAM error MSI writes.
 MPAMF_ERR_MSI_MPAM_rt controls MPAM information labeling of Root MPAM error MSI writes.
 MPAMF_ERR_MSI_MPAM_rl controls MPAM information labeling of Realm MPAM error MSI writes.

Configuration

The power domain of MPAMF_ERR_MSI_MPAM is IMPLEMENTATION DEFINED.

This register is present only when MPAMF_IDR.HAS_ERR_MSI == 1. Otherwise, direct accesses to MPAMF_ERR_MSI_MPAM are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMF_ERR_MSI_MPAM 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								PARTID															

Bits [31:24]

Reserved, RES0.

PMG, bits [23:16]

Performance monitoring group property for PARTID MSC error interrupt write.

The reset behavior of this field is:

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

PARTID, bits [15:0]

Partition ID for MSC error interrupt write.

The PARTID in this register is in the Secure PARTID space in the MPAMF_ERR_MSI_MPAM_s instance and in the Non-secure PARTID space in the MPAMF_ERR_MSI_MPAM_ns instance of this register.

The reset behavior of this field is:

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

Accessing MPAMF_ERR_MSI_MPAM

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:maps.~~

- MPAMF_ERR_MSI_MPAM_s must be accessible from the Secure MPAM feature page.
- MPAMF_ERR_MSI_MPAM_ns must be accessible from the Non-secure MPAM feature page.
- MPAMF_ERR_MSI_MPAM_rt must be accessible from the Root MPAM feature page.
- MPAMF_ERR_MSI_MPAM_rl must be accessible from the Realm MPAM feature page.

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:registers.~~

- The Secure instance (MPAMF_ERR_MSI_MPAM_s) accesses the MPAM information for MSI write request to signal an MPAM error used for Secure PARTIDs.
- The Non-secure instance (MPAMF_ERR_MSI_MPAM_ns) accesses the MPAM information for MSI write request to signal an MPAM error used for Non-secure PARTIDs.
- The Root instance (MPAMF_ERR_MSI_MPAM_rt) accesses the MPAM information for MSI write request to signal an MPAM error used for Root PARTIDs.
- The Realm instance (MPAMF_ERR_MSI_MPAM_rl) accesses the MPAM information for MSI write request to signal an MPAM error used for Realm PARTIDs.

MPAMF_ERR_MSI_MPAM can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x00DC	MPAMF_ERR_MSI_MPAM_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x00DC	MPAMF_ERR_MSI_MPAM_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x00DC	MPAMF_ERR_MSI_MPAM_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x00DC	MPAMF_ERR_MSI_MPAM_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

MPAMF_ESR, MPAM Error Status Register

The MPAMF_ESR characteristics are:

Purpose

Indicates MPAM error status for this MSC.

MPAMF_ESR_s reports Secure MPAM errors. MPAMF_ESR_ns reports Non-secure MPAM errors. MPAMF_ESR_rt reports Root MPAM errors. MPAMF_ESR_rl reports Realm MPAM errors.

Software should write this register after reading the status of an error to reset ERRCODE to 0x0000 and OVRWR to 0 so that future errors are not reported with OVRWR set.

Configuration

The power domain of MPAMF_ESR is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented. Otherwise, direct accesses to MPAMF_ESR are RES0.

MPAMF_ESR is 64-bit register when MPAM v0.1 or v1.1 is implemented and MPAMF_IDR.HAS_EXTD_ESR == 1.

Otherwise, MPAMF_ESR is a 32-bit register.

If an MSC cannot encounter any of the error conditions listed in 'Errors in MSCs' in Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A (ARM DDI 0598), both the MPAMF_ESR and [MPAMF_ECR](#) must be RAZ/WI.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMF_ESR is a:

- 64-bit register when (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_IDR.HAS_EXTD_ESR == 1
- 32-bit register otherwise

Field descriptions

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_IDR.HAS_EXTD_ESR == 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																															RIS	
OVRWR	RES0			ERRCODE				PMG								PARTID_MON																
31	30	29	28	27	26	25	24	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 [63:36]

Reserved, RES0.

RIS, bits [35:32]

When MPAMF_IDR.HAS_RIS == 1:

Resource Instance Selector. Where applicable to the ERRCODE, captures the RIS value for the error.

Otherwise:

Reserved, RES0.

OVRWR, bit [31]

Overwritten.

If 0 and ERRCODE == 0b0000, no errors have occurred.

If 0 and ERRCODE is non-zero, a single error has occurred and is recorded in this register.

If 1 and ERRCODE is non-zero, multiple errors have occurred and this register records the most recent error.

The state where this bit is 1 and ERRCODE is zero must not be produced by hardware and is only reached when software writes this combination into this register.

Bits [30:28]

Reserved, RES0.

ERRCODE, bits [27:24]

Error code.

ERRCODE	Meaning
0b0000	No error.
0b0001	PARTID_SEL_Range.
0b0010	Req_PARTID_Range.
0b0011	MSMONCFG_ID_RANGE.
0b0100	Req_PMG_Range.
0b0101	Monitor_Range.
0b0110	intPARTID_Range.
0b0111	Unexpected_INTERNAL.
0b1000	Undefined_RIS_PART_SEL.
0b1001	RIS_No_Control.
0b1010	Undefined_RIS_MON_SEL.
0b1011	RIS_No_Monitor.
0b1100	Reserved.
0b1101	Reserved.
0b1110	Reserved.
0b1111	Reserved.

PMG, bits [23:16]

Program monitoring group.

Set to the PMG on an error that captures PMG. Otherwise, set to 0x00 on an error that does not capture PMG.

PARTID_MON, bits [15:0]

PARTID or monitor.

Set to the PARTID on an error that captures PARTID.

Set to the monitor index on an error that captures MON.

On an error that captures neither PARTID nor MON, this field is set to 0.

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
OVRWR		RES0		ERRCODE			PMG							PARTID MON																	

OVRRWR, bit [31]

Overwritten.

If 0 and ERRCODE == 0b0000, no errors have occurred.

If 0 and ERRCODE is non-zero, a single error has occurred and is recorded in this register.

If 1 and ERRCODE is non-zero, multiple errors have occurred and this register records the most recent error.

The state where this bit is 1 and ERRCODE is 0 must not be produced by hardware and is only reached when software writes this combination into this register.

Bits [30:28]

Reserved, RES0.

ERRCODE, bits [27:24]

Error code.

ERRCODE	Meaning
0b0000	No error.
0b0001	PARTID_SEL_Range.
0b0010	Req_PARTID_Range.
0b0011	MSMONCFG_ID_RANGE.
0b0100	Req_PMG_Range.
0b0101	Monitor_Range.
0b0110	intPARTID_Range.
0b0111	Unexpected_INTERNAL.
0b1000	Reserved.
0b1001	Reserved.
0b1010	Reserved.
0b1011	Reserved.
0b1100	Reserved.
0b1101	Reserved.
0b1110	Reserved.
0b1111	Reserved.

PMG, bits [23:16]

Program monitoring group.

Set to the PMG on an error that captures PMG. Otherwise, set to 0x00 on an error that does not capture PMG.

PARTID_MON, bits [15:0]

PARTID or monitor.

Set to the PARTID on an error that captures PARTID.

Set to the monitor index on an error that captures MON.

On an error that captures neither PARTID nor MON, this field is set to 0x0000.

Accessing MPAMF_ESR

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:maps**.

- MPAMF_ESR_s must be accessible from the Secure MPAM feature page.
- MPAMF_ESR_ns must be accessible from the Non-secure MPAM feature page.
- MPAMF_ESR_rt must be accessible from the Root MPAM feature page.

- MPAMF_ESR_rl must be accessible from the Realm MPAM feature page.

MPAMF_ESR_s, MPAMF_ESR_ns, MPAMF_ESR_rt, and MPAMF_ESR_rl must be separate registers: registers.

- The Secure instance (MPAMF_ESR_s) accesses the error status used for Secure PARTIDs.
- The Non-secure instance (MPAMF_ESR_ns) accesses the error status used for Non-secure PARTIDs.
- The Root instance (MPAMF_ESR_rt) accesses the error status used for Root PARTIDs.
- The Realm instance (MPAMF_ESR_rl) accesses the error status used for Realm PARTIDs.

MPAMF_ESR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x00F8	MPAMF_ESR_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x00F8	MPAMF_ESR_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x00F8	MPAMF_ESR_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x00F8	MPAMF_ESR_rl

When FEAT_RME is implemented access on this interface are **RW**.

30/09/2021 14:53:37; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MPAMF_IDR, MPAM Features Identification Register

The MPAMF_IDR characteristics are:

Purpose

Indicates which memory partitioning and monitoring features are present on this MSC.

MPAMF_IDR_s indicates the MPAM features accessed from the Secure MPAM feature page. MPAMF_IDR_ns indicates the MPAM features accessed from the Non-secure MPAM feature page. MPAMF_IDR_rt indicates the MPAM features accessed from the Root MPAM feature page. MPAMF_IDR_rl indicates the MPAM features accessed from the Realm MPAM feature page.

When MPAMF_IDR.HAS_RIS is 1, some fields in this register give information for the resource instance selected by [MPAMCFG_PART_SEL](#).RIS. The description of every field that is affected by [MPAMCFG_PART_SEL](#).RIS has that information within the field description.

Configuration

The power domain of MPAMF_IDR is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented. Otherwise, direct accesses to MPAMF_IDR are RES0.

MPAMF_IDR is 64-bit register when MPAM v0.1 or v1.1 is implemented.

Otherwise, MPAMF_IDR is a 32-bit register.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMF_IDR is a:

- 64-bit register when FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented
- 32-bit register otherwise

Field descriptions

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

63	62	61	60	59	58	57	56	55
RES0				RIS_MAX				
HAS_PARTID_NRW	HAS_MSMON	HAS_IMPL_IDR_EXT	HAS_PRI_PART	HAS_MBW_PART	HAS_CPOR_PART	HAS_CCAP_PART		
31	30	29	28	27	26	25	24	23

Bits [63:60]

Reserved, RES0.

RIS_MAX, bits [59:56]

When MPAMF_IDR.EXT == 1 and MPAMF_IDR.HAS_RIS == 1:

Maximum RIS value supported in [MPAMCFG_PART_SEL](#). Must be 0b0000 if [MPAMF_IDR](#).HAS_RIS == 0.

Otherwise:

Reserved, RES0.

Bits [55:44]

Reserved, RES0.

HAS_NFU, bit [43]**When FEAT_MPAMv1p1 is implemented or FEAT_MPAMv0p1 is implemented:**Has No Future Use field in [MPAMCFG_DIS](#). Indicates that [MPAMCFG_DIS.NFU](#) is implemented.

HAS_NFU	Meaning
0b0	MPAMCFG_DIS.NFU is not implemented. A PARTID disabled through access to MPAMCFG_DIS must preserve the control settings of the disabled PARTID.
0b1	Implements MPAMCFG_DIS.NFU . A PARTID disabled with NFU as 1 may have its control settings forgotten.

If [MPAMF_IDR.HAS_ENDIS](#) is 0b0, this field must also be 0b0.This field must be the same in each instance of this register and for any value in [MPAMCFG_PART_SEL.RIS](#).**Otherwise:**

Reserved, RES0.

HAS_ENDIS, bit [42]**When FEAT_MPAMv1p1 is implemented or FEAT_MPAMv0p1 is implemented:**Has PARTID enable and disable. Indicates that this MSC supports PARTID disable and enable via [MPAMCFG_DIS](#), [MPAMCFG_EN](#) and [MPAMCFG_EN_FLAGS](#) registers.

HAS_ENDIS	Meaning
0b0	Does not support PARTID enable and disable functionality, and MPAMCFG_EN , MPAMCFG_DIS and MPAMCFG_EN_FLAGS registers are not implemented.
0b1	Supports PARTID enable and disable through the MPAMCFG_EN , MPAMCFG_DIS and MPAMCFG_EN_FLAGS registers.

All three registers must be implemented when this field is 1, [MPAMCFG_EN](#), [MPAMCFG_DIS](#), and [MPAMCFG_EN_FLAGS](#).This field must be the same in each instance of this register and for any value in [MPAMCFG_PART_SEL.RIS](#).**Otherwise:**

Reserved, RES0.

SP4, bit [41]**When FEAT_RME is implemented:**

Indicates whether this MSC supports 4 PARTID spaces.

SP4	Meaning
0b0	This MSC supports two PARTID spaces.
0b1	This MSC supports four PARTID spaces.

This field must read the same in each instance of this register and for any value in [MPAMCFG_PART_SEL.RIS](#).

Otherwise:

Reserved, RES0.

HAS_ERR_MSI, bit [40]

When `MPAMF_IDR.EXT == 1`:

Has support for MSI writes to signal MPAM error interrupts. These registers are implemented: [MPAMF_ERR_MSI_ADDR_L](#), [MPAMF_ERR_MSI_ADDR_H](#), [MPAMF_ERR_MSI_ATTR](#), [MPAMF_ERR_MSI_DATA](#), and [MPAMF_ERR_MSI_MPAM](#).

HAS_ERR_MSI	Meaning
0b0	MPAMF_ERR_MSI_ADDR_L , MPAMF_ERR_MSI_ADDR_H , MPAMF_ERR_MSI_ATTR , MPAMF_ERR_MSI_DATA , and MPAMF_ERR_MSI_MPAM registers are not implemented.
0b1	MPAMF_ERR_MSI_ADDR_L , MPAMF_ERR_MSI_ADDR_H , MPAMF_ERR_MSI_ATTR , MPAMF_ERR_MSI_DATA , and MPAMF_ERR_MSI_MPAM are implemented and can be used to generate writes to signal error interrupts.

If [MPAMF_IDR.HAS_ESR](#) is 0, this bit must also be 0.

Otherwise:

Reserved, RES0.

HAS_ESR, bit [39]

When `MPAMF_IDR.EXT == 1`:

[MPAMF_ESR](#) is implemented.

HAS_ESR	Meaning
0b0	MPAMF_ESR , MPAMF_ECR , and MPAM error handling are not implemented.
0b1	MPAMF_ESR , MPAMF_ECR , and MPAM error handling are implemented.

If an MSC cannot encounter any of the error conditions listed in 'Errors in MSCs' in Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A (ARM DDI 0598), both the [MPAMF_ESR](#) and [MPAMF_ECR](#) must be RAZ/WI.

Otherwise:

Reserved, RES0.

HAS_EXTD_ESR, bit [38]

When `MPAMF_IDR.EXT == 1`:

[MPAMF_ESR](#) is 64 bits.

HAS_EXTD_ESR	Meaning
0b0	MPAMF_ESR is 32 bits.
0b1	MPAMF_ESR is 64 bits.

When [MPAMF_IDR.HAS_RIS](#) and [MPAMF_IDR.HAS_ESR](#), this field must be 1.

Otherwise:

Reserved, RES0.

NO_IMPL_MSMON, bit [37]

When MPAMF_IDR.EXT == 1 and MPAMF_IDR.HAS_IMPL_IDR == 1:

[MPAMF_IMPL_IDR](#) defines no IMPLEMENTATION DEFINED resource monitors.

NO_IMPL_MSMON	Meaning
0b0	MPAMF_IMPL_IDR defines at least one IMPLEMENTATION DEFINED resource monitor.
0b1	MPAMF_IMPL_IDR does not define any IMPLEMENTATION DEFINED resource monitors.

If RIS is implemented, this field indicates the presence of IMPLEMENTATION DEFINED resource monitors described in [MPAMF_IMPL_IDR](#) for the selected resource instance.

Otherwise:

Reserved, RES0.

NO_IMPL_PART, bit [36]

When MPAMF_IDR.EXT == 1 and MPAMF_IDR.HAS_IMPL_IDR == 1:

[MPAMF_IMPL_IDR](#) defines no IMPLEMENTATION DEFINED resource controls.

NO_IMPL_PART	Meaning
0b0	MPAMF_IMPL_IDR defines at least one IMPLEMENTATION DEFINED resource control.
0b1	MPAMF_IMPL_IDR does not define any IMPLEMENTATION DEFINED resource controls.

If RIS is implemented, this field indicates the presence of IMPLEMENTATION DEFINED resource controls described in [MPAMF_IMPL_IDR](#) for the selected resource instance.

Otherwise:

Reserved, RES0.

Bits [35:33]

Reserved, RES0.

HAS_RIS, bit [32]

When MPAMF_IDR.EXT == 1:

Has resource instance selector. Indicates that [MPAMCFG_PART_SEL](#) contains the RIS field that selects a resource instance to control.

HAS_RIS	Meaning
0b0	MPAMCFG_PART_SEL does not implement the MPAMCFG_PART_SEL .RIS field or multiple resource instance support.
0b1	MPAMCFG_PART_SEL implements the MPAMCFG_PART_SEL .RIS field and MPAM resource instance numbers up to and including MPAMF_IDR.RIS_MAX.

Otherwise:

Reserved, RES0.

HAS_PARTID_NRW, bit [31]

Has PARTID narrowing.

HAS_PARTID_NRW	Meaning
0b0	Does not have MPAMF_PARTID_NRW_IDR , MPAMCFG_INTPARTID , or intPARTID mapping support.
0b1	Supports the MPAMF_PARTID_NRW_IDR , MPAMCFG_INTPARTID registers.

HAS_MSMON, bit [30]

Has resource [Monitors](#). ~~monitors~~. Indicates whether this MSC has MPAM resource monitors.

HAS_MSMON	Meaning
0b0	Does not support MPAM resource monitoring by groups or MPAMF_MSMON_IDR .
0b1	Supports resource monitoring by matching a combination of PARTID and PMG. See MPAMF_MSMON_IDR .

HAS_IMPL_IDR, bit [29]

Has [MPAMF_IMPL_IDR](#). Indicates whether this MSC has the IMPLEMENTATION SPECIFIC MPAM features register, [MPAMF_IMPL_IDR](#).

HAS_IMPL_IDR	Meaning
0b0	Does not have MPAMF_IMPL_IDR .
0b1	Has MPAMF_IMPL_IDR .

EXT, bit [28]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

Extended MPAMF_IDR.

EXT	Meaning
0b0	MPAMF_IDR has no defined bits in [63:32]. The register is effectively 32 bits.
0b1	MPAMF_IDR has bits defined in [63:32]. The register is 64-bits.

Otherwise:

Reserved, RES0.

HAS_PRI_PART, bit [27]

Has [Priority](#) ~~priority~~ [Partitioning](#). ~~partitioning~~. Indicates that MPAM priority partitioning is implemented and [MPAMF_PRI_IDR](#) exists.

HAS_PRI_PART	Meaning
0b0	Does not support priority partitioning or have MPAMF_PRI_IDR .
0b1	Has priority partitioning and MPAMF_PRI_IDR .

If RIS is implemented, this field indicates the presence of priority partitioning resource controls as described in [MPAMF_PRI_IDR](#) for the selected resource instance.

HAS_MBW_PART, bit [26]

Has **Memory** ~~memory~~ **Bandwidth** ~~bandwidth~~ **Partitioning** ~~partitioning~~. Indicates whether this MSC implements MPAM memory bandwidth partitioning and [MPAMF_MBW_IDR](#).

HAS_MBW_PART	Meaning
0b0	Does not support memory bandwidth partitioning or have MPAMF_MBW_IDR register.
0b1	Has MPAMF_MBW_IDR register.

If RIS is implemented, this field indicates the presence of memory bandwidth partitioning resource controls as described in [MPAMF_MBW_IDR](#) for the selected resource instance.

HAS_CPOR_PART, bit [25]

Has **Cache** ~~cache~~ **Portion** ~~portion~~ **Partitioning** ~~partitioning~~. Indicates whether this MSC implements MPAM cache portion partitioning and [MPAMF_CPOR_IDR](#).

HAS_CPOR_PART	Meaning
0b0	Does not support cache portion partitioning or have MPAMF_CPOR_IDR or MPAMCFG_CPB<n> registers.
0b1	Has MPAMF_CPOR_IDR and MPAMCFG_CPB<n> registers.

If RIS is implemented, this field indicates the presence of cache portion partitioning resource controls as described in [MPAMF_CPOR_IDR](#) for the selected resource instance.

HAS_CCAP_PART, bit [24]

Has **Cache** ~~cache~~ **Capacity** ~~capacity~~ **Partitioning** ~~partitioning~~. Indicates whether this MSC implements MPAM cache capacity partitioning and the [MPAMF_CCAP_IDR](#) and [MPAMCFG_CMAX](#) registers.

HAS_CCAP_PART	Meaning
0b0	Does not support cache capacity partitioning or have MPAMF_CCAP_IDR and MPAMCFG_CMAX registers.
0b1	Has MPAMF_CCAP_IDR and MPAMCFG_CMAX registers.

If RIS is implemented, this field indicates the presence of cache capacity partitioning resource controls as described in [MPAMF_CPOR_IDR](#) for the selected resource instance.

PMG_MAX, bits [23:16]

Maximum **supported** value of **PMG**. ~~Non-secure PMG supported by this component.~~

The value of this field is permitted to vary between the instances of [MPAM_IDR](#), each reporting the maximum supported PMG value in the PARTID space associated with that instance.

In [MPAMF_IDR](#) s, this field is permitted to report the maximum PMG value for the Non-secure PARTID space or for the Secure PARTID space. The maximum PMG value for the Secure PARTID space can be read from [MPAMF_SIDR.PMG_MAX](#).

PARTID_MAX, bits [15:0]

Maximum **supported** value of **PARTID**. ~~Non-secure PARTID supported by this component.~~

The value of this field is permitted to vary between the instances of [MPAM_IDR](#), each reporting the maximum supported PARTID value in the PARTID space associated with that instance.

In [MPAMF_IDR](#) s, this field is permitted to report the maximum PARTID value for the Non-secure PARTID space or for the Secure PARTID space. The maximum PARTID value for the Secure PARTID space can be read from [MPAMF_SIDR.PARTID_MAX](#).

Otherwise:

31	30	29	28	27	26	25	24	23
HAS_PARTID_NRW	HAS_MSMON	HAS_IMPL_IDR	EXT	HAS_PRI_PART	HAS_MBW_PART	HAS_CPOR_PART	HAS_CCAP_PART	

HAS_PARTID_NRW, bit [31]

Has PARTID **Narrowing**~~narrowing~~.

HAS_PARTID_NRW	Meaning
0b0	Does not have MPAMF_PARTID_NRW_IDR , MPAMCFG_INTPARTID , or intPARTID mapping support.
0b1	Supports the MPAMF_PARTID_NRW_IDR , MPAMCFG_INTPARTID registers.

HAS_MSMON, bit [30]

Has resource **Monitors**~~monitors~~. Indicates whether this MSC has MPAM resource monitors.

HAS_MSMON	Meaning
0b0	Does not support MPAM resource monitoring by groups or MPAMF_MSMON_IDR .
0b1	Supports resource monitoring by matching a combination of PARTID and PMG. See MPAMF_MSMON_IDR .

HAS_IMPL_IDR, bit [29]

Has [MPAMF_IMPL_IDR](#). Indicates whether this MSC has the IMPLEMENTATION SPECIFIC MPAM features register, [MPAMF_IMPL_IDR](#).

HAS_IMPL_IDR	Meaning
0b0	Does not have MPAMF_IMPL_IDR .
0b1	Has MPAMF_IMPL_IDR .

EXT, bit [28]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

Extended MPAMF_IDR.

EXT	Meaning
0b0	MPAMF_IDR has no defined bits in [63:32]. The register is effectively 32 bits.
0b1	MPAMF_IDR has bits defined in [63:32]. The register is 64-bits.

Otherwise:

Reserved, RES0.

HAS_PRI_PART, bit [27]

Has **Priority**~~priority~~ **Partitioning**~~partitioning~~. Indicates whether this MSC implements MPAM priority partitioning and [MPAMF_PRI_IDR](#).

HAS_PRI_PART	Meaning
0b0	Does not support priority partitioning or have MPAMF_PRI_IDR .
0b1	Has MPAMF_PRI_IDR .

HAS_MBW_PART, bit [26]

Has ~~Memory~~memory ~~Bandwidth~~bandwidth ~~Partitioning~~partitioning. Indicates whether this MSC implements MPAM memory bandwidth partitioning and MPAMF_MBW_IDR.

HAS_MBW_PART	Meaning
0b0	Does not support memory bandwidth partitioning or have MPAMF_MBW_IDR register.
0b1	Has MPAMF_MBW_IDR register.

HAS_CPOR_PART, bit [25]

Has ~~Cache~~cache ~~Portion~~portion ~~Partitioning~~partitioning. Indicates whether this MSC implements MPAM cache portion partitioning and [MPAMF_CPOR_IDR](#).

HAS_CPOR_PART	Meaning
0b0	Does not support cache portion partitioning or have MPAMF_CPOR_IDR or MPAMCFG_CPBm<n> registers.
0b1	Has MPAMF_CPOR_IDR and MPAMCFG_CPBm<n> registers.

HAS_CCAP_PART, bit [24]

Has ~~Cache~~cache ~~Capacity~~capacity ~~Partitioning~~partitioning. Indicates whether this MSC implements MPAM cache capacity partitioning and the MPAMF_CCAP_IDR and MPAMCFG_CMAX registers.

HAS_CCAP_PART	Meaning
0b0	Does not support cache capacity partitioning or have MPAMF_CCAP_IDR and MPAMCFG_CMAX registers.
0b1	Has MPAMF_CCAP_IDR and MPAMCFG_CMAX registers.

PMG_MAX, bits [23:16]

Maximum [supported](#) value of ~~PMG~~~~Non-secure PMG supported by this component.~~

The value of this field is permitted to vary between the instances of [MPAM_IDR](#), each reporting the maximum supported PMG value in the PARTID space associated with that instance.

In MPAMF_IDR s this field is permitted to report the maximum PMG value for the Non-secure PARTID space or for the Secure PARTID space. The maximum PMG value for the Secure PARTID space can be read from [MPAMF_SIDR.PMG_MAX](#).

PARTID_MAX, bits [15:0]

Maximum [supported](#) value of ~~PARTID~~~~Non-secure PARTID supported by this component.~~

The value of this field is permitted to vary between the instances of [MPAM_IDR](#), each reporting the maximum supported PARTID value in the PARTID space associated with that instance.

In MPAMF_IDR s this field is permitted to report the maximum PARTID value for the Non-secure PARTID space or for the Secure PARTID space. The maximum PARTID value for the Secure PARTID space can be read from [MPAMF_SIDR.PARTID_MAX](#).

Accessing MPAMF_IDR

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_IDR is read-only.

MPAMF_IDR must be readable from the Non-secure, Secure, Root, and Realm MPAM feature pages.

MPAMF_IDR is permitted to have the same contents when read from the Secure, Non-secure, Root, and Realm MPAM feature pages unless the register contents are different for the different versions:

- MPAMF_IDR_s is permitted to have either the same or different contents to MPAMF_IDR_ns, MPAMF_IDR_rt, or MPAMF_IDR_rl.
- MPAMF_IDR_ns is permitted to have either the same or different contents to MPAMF_IDR_rt or MPAMF_IDR_rl.
- MPAMF_IDR_rt is permitted to have either the same or different contents to MPAMF_IDR_rl.

There must be separate registers in the Secure (MPAMF_IDR_s), Non-secure (MPAMF_IDR_ns), Root (MPAMF_IDR_rt), and Realm (MPAMF_IDR_rl) MPAM feature pages.

When [MPAMF_IDR.HAS_RIS](#) is 1, MPAMF_IDR shows the configuration of MSC MPAM for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#). Fields that mention RIS in their field descriptions have values that track the implemented properties of the resource instance. Fields that do not mention RIS are constant across all resource instances.

MPAMF_IDR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0000	MPAMF_IDR_s

Accesses on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0000	MPAMF_IDR_ns

Accesses on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0000	MPAMF_IDR_rt

When FEAT_RME is implemented access on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0000	MPAMF_IDR_rl

When FEAT_RME is implemented access on this interface are **RO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbf36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MPAMF_MBWUMON_IDR, MPAM Features Memory Bandwidth Usage Monitoring ID register

The MPAMF_MBWUMON_IDR characteristics are:

Purpose

Indicates the number of memory bandwidth usage monitor instances implemented. This register also indicates several properties of MBWU monitoring, including whether the implementation supports capture, scaling, or long counters.

MPAMF_MBWUMON_IDR_s indicates the number of Secure memory bandwidth usage monitor instances.

MPAMF_MBWUMON_IDR_ns indicates the number of Non-secure memory bandwidth usage monitor instances.

MPAMF_MBWUMON_IDR_rt indicates the number of Root memory bandwidth usage monitor instances.

MPAMF_MBWUMON_IDR_rl indicates the number of Realm memory bandwidth usage monitor instances.

If [MPAMF_IDR.HAS_RIS](#) is 1, fields that mention RIS must reflect the properties of the resource instance currently selected by [MPAMCFG_PART_SEL.RIS](#). Fields that do not mention RIS are constant across all resource instances.

Configuration

The power domain of MPAMF_MBWUMON_IDR 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 MPAMF_MBWUMON_IDR are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MPAMF_MBWUMON_IDR is a 32-bit register.

Field descriptions

31	30	29	28	27	26	25	24	23	22	21	20
HAS_CAPTURE	HAS_LONG	LWD	HAS_RWBW	HAS_OFLOW_LNKG	RES0	HAS_OFSR	HAS_OFLOW_CAPT	RES0	RES0	SCALE	SC

HAS_CAPTURE, bit [31]

The implementation supports copying an [MSMON_MBWU](#) to the corresponding [MSMON_MBWU_CAPTURE](#) on a capture event.

HAS_CAPTURE	Meaning
0b0	MSMON_MBWU_CAPTURE is not implemented and there is no support for capture events in the MBWU monitor.
0b1	The MSMON_MBWU_CAPTURE register is implemented and the MBWU monitor supports the capture event behavior.

If RIS is implemented, this field indicates that MBWU monitor capture is implemented for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#).

If MPAMF_MBWUMON_IDR.HAS_LONG is 1, this also indicates that [MSMON_MBWU_L_CAPTURE](#) is implemented.

HAS_LONG, bit [30]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

Indicates whether [MSMON_MBWU_L](#) is implemented.

If HAS_CAPTURE is 1, indicates whether [MSMON_MBWU_L_CAPTURE](#) is implemented.

HAS_LONG	Meaning
0b0	Does not implement MSMON_MBWU_L or MSMON_MBWU_L_CAPTURE .
0b1	Implements MSMON_MBWU_L . If HAS_CAPTURE == 1, MSMON_MBWU_L_CAPTURE is also implemented.

If RIS is implemented, this field indicates that the long MBWU monitor is implemented for the resource instance selected by [MPAMCFG_PART_SEL](#).RIS.

If MPAMF_MBWUMON_IDR.HAS_CAPTURE is 1, this also indicates that [MSMON_MBWU_L_CAPTURE](#) is implemented.

Otherwise:

Reserved, RES0.

LWD, bit [29]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

Long register VALUE width.

If [MPAMF_MBWUMON_IDR](#).HAS_LONG is 0, [MPAMF_MBWUMON_IDR](#).LWD must also be 0.

LWD	Meaning
0b0	If MPAMF_MBWUMON_IDR .HAS_LONG is 1, MSMON_MBWU_L has 44-bit VALUE field in bits [43:0]. Bits [62:44] are RES0. If HAS_LONG is 1 and MPAMF_MBWUMON_IDR .HAS_CAPTURE is 1, MSMON_MBWU_L_CAPTURE also has 44-bit VALUE field in bits [43:0].
0b1	MSMON_MBWU_L has 63-bit VALUE field in bits [62:0]. If MPAMF_MBWUMON_IDR .HAS_CAPTURE == 1, MSMON_MBWU_L_CAPTURE also has 63-bit VALUE field in bits [62:0].

If RIS is implemented, this field indicates the length of the [MSMON_MBWU_L](#).VALUE field implemented for the resource instance selected by [MPAMCFG_PART_SEL](#).RIS.

Otherwise:

Reserved, RES0.

HAS_RWBW, bit [28]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

Read/write bandwidth selection is implemented in [MSMON_CFG_MBWU_FLT](#).

HAS_RWBW	Meaning
0b0	Read/write bandwidth selection is not implemented.
0b1	Read/write bandwidth selection is implemented.

If RIS is implemented, this field indicates whether read/write bandwidth collection selection is available in [MSMON_CFG_MBWU_FLT](#) for resource instance selected by [MPAMCFG_PART_SEL](#).RIS.

Otherwise:

Reserved, RES0.

HAS_OFLOW_LNKG, bit [27]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

Supports [MSMON_CFG_MBWU_CTL.OFLOW_LNKG](#) field to control how overflow on an instance affects other monitor instances in this MSC.

HAS_OFLOW_LNKG	Meaning
0b0	Does not support MBWU overflow linkage.
0b1	Supports MBWU overflow linkage and the MSMON_CFG_MBWU_CTL.OFLOW_LNKG field.

If RIS is implemented, this field indicates that [MSMON_CFG_MBWU_CTL.OFLOW_LNKG](#) is implemented for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#).

Otherwise:

Reserved, RES0.

HAS_OFSR, bit [26]

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented:

The MBWU monitor overflow status bitmap register, [MSMON_MBWU_OFSR](#), is implemented.

HAS_OFSR	Meaning
0b0	MSMON_MBWU_OFSR register is not implemented.
0b1	MSMON_MBWU_OFSR register is implemented.

If RIS is implemented, this field indicates that MBWU monitor overflow status bitmap register is implemented for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#).

Otherwise:

Reserved, RES0.

HAS_OFLOW_CAPT, Bits [25:21]

Supports [MSMON_CFG_MBWU_CTL.OFLOW_CAPT](#) field to transfer the MBWU monitor instance to its capture register on an overflow or overflow linkage event.

HAS_OFLOW_CAPT	Meaning
0b0	Does not support MBWU capture on overflow.
0b1	Supports MBWU capture on overflow and the MSMON_CFG_MBWU_CTL.OFLOW_CAPT field.

If RIS is implemented, this field indicates that [MSMON_CFG_MBWU_CTL.OFLOW_CAPT](#) is implemented for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#).

Bits [24:21]

Reserved, RES0.

SCALE, bits [20:16]

Scaling of [MSMON_MBWU.VALUE](#) in bits. If scaling is enabled by [MSMON_CFG_MBWU_CTL.SCLEN](#), the byte count in the VALUE field has been shifted by SCALE bits to the right.

SCALE	Meaning
0b00000	Scaling is not implemented.
0bxxxxx	Other values are right shift count when scaling is enabled.

If RIS is implemented, this field indicates the scale value for [MSMON_MBWU.VALUE](#) field for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#).

NUM_MON, bits [15:0]

The number of memory bandwidth usage monitor instances implemented. The largest monitor instance selector, [MSMON_CFG_MON_SEL.MON_SEL](#), is NUM_MON minus 1.

If RIS is implemented, this field indicates the number of MBWU monitor instances for [MSMON_MBWU.VALUE](#) field for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#).

Accessing MPAMF_MBWUMON_IDR

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_MBWUMON_IDR is read-only.

MPAMF_MBWUMON_IDR must be readable from the Non-secure, Secure, Root, and Realm MPAM feature pages.

MPAMF_MBWUMON_IDR is permitted to have the same contents when read from the Secure, Non-secure, Root, and Realm MPAM feature pages unless the register contents are different for the different versions:

- MPAMF_MBWUMON_IDR_s is permitted to have either the same or different contents to MPAMF_MBWUMON_IDR_ns, MPAMF_MBWUMON_IDR_rt, or MPAMF_MBWUMON_IDR_rl.
- MPAMF_MBWUMON_IDR_ns is permitted to have either the same or different contents to MPAMF_MBWUMON_IDR_rt or MPAMF_MBWUMON_IDR_rl.
- MPAMF_MBWUMON_IDR_rt is permitted to have either the same or different contents to MPAMF_MBWUMON_IDR_rl.

There must be separate registers in the Secure (MPAMF_MBWUMON_IDR_s), Non-secure (MPAMF_MBWUMON_IDR_ns), Root (MPAMF_MBWUMON_IDR_rt), and Realm (MPAMF_MBWUMON_IDR_rl) MPAM feature pages.

When [MPAMF_IDR.HAS_RIS](#) is 1, MPAMF_MBWUMON_IDR shows the configuration of memory bandwidth monitoring for the bandwidth resource instance selected by [MPAMCFG_PART_SEL.RIS](#). Fields that mention RIS in their field descriptions have values that track the implemented properties of the resource instance. Fields that do not mention RIS are constant across all resource instances.

Access to MPAMF_MBWUMON_IDR is not affected by [MSMON_CFG_MON_SEL.RIS](#).

MPAMF_MBWUMON_IDR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0090	MPAMF_MBWUMON_IDR_s

Accesses on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0090	MPAMF_MBWUMON_IDR_ns

Accesses on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0090	MPAMF_MBWUMON_IDR_rt

When FEAT_RME is implemented access on this interface are **RO**.

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(new)

The MSMON CFG CSU CTL characteristics are:

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`.

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.

MSMON_CFG_CSU_CTL is a 32-bit register.

31	30	29	28	27	26	25	24	23	22	21	20	19
EN	CAPT_EVNT	CAPT_RESET	OFLOW_STATUS	OFLOW_INTR	OFLOW_FRZ	OFLOW_CAPT	SUBTYPE	SUBTYPE_RES0	RES0_MAT	RES1_MAT	RES2_MAT	RES3_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.

If capture is not implemented for the CSU monitor type as indicated by [MPAMF_CSUMON_IDR.HAS_CAPTURE](#) = 0, 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.

If capture is not implemented for the CSU monitor type as indicated by [MPAMF_CSUMON_IDR.HAS_CAPTURE](#) = 0, this field is RAZ/WI.

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.

OFLOW_STATUS, bit [26]

Overflow status.

Indicates whether the value of [MSMON_CSU](#) has overflowed.

If [MPAMF_CSUMON_IDR.HAS_OFLOW_CAPT](#) 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.

If OFLOW_INTR is 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](#) 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.

OFLOW_CAPT **SUBTYPE, bit** **bits [23:20]**

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_CSUMON_IDR.HAS_OFLOW_CAPT == 1:

Capture Monitor on Overflow.

Controls whether the value of [MSMON_CSU](#) is captured on an overflow or an overflow linkage event.

OFLOW_CAPT	Meaning
0b0	Monitor is not captured on an overflow or when affected by an overflow linkage event.
0b1	Monitor 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 RIS is implemented, this field indicates that [MSMON_CFG_CSU_CTL.OFLOW_CAPT](#) is implemented for the resource instance selected by [MPAMCFG_PART_SEL.RIS](#).

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 **Bits [19:18]**

Reserved, RES0.

CEVNT_OFLW, bit [18]

When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_CSUMON_IDR.HAS_OFLOW_LNKG == 1:

Capture Event performs overflow behavior.

Selects whether a capture event matching the CAPT_EVNT field performs the overflow behavior or the capture behavior.

CEVNT_OFLW	Meaning
0b0	On a capture event matching the CAPT_EVNT field, the capture behaviors are performed.
0b1	On a capture event matching the CAPT_EVNT field, the overflow behaviors are performed.

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 [15:118]

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:maps~~.

- MSMON_CFG_CSU_CTL_s must be accessible from the Secure MPAM feature page.
- MSMON_CFG_CSU_CTL_ns must be accessible from the Non-secure MPAM feature page.
- MSMON_CFG_CSU_CTL_rt must be accessible from the Root MPAM feature page.
- MSMON_CFG_CSU_CTL_rl must 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: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.

MSMON_CFG_CSU_CTL can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0818	MSMON_CFG_CSU_CTL_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0818	MSMON_CFG_CSU_CTL_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0818	MSMON_CFG_CSU_CTL_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
-----------	-------	--------	----------

MPAM	MPAMF_BASE_r1	0x0818	MSMON_CFG_CSU_CTL_r1
------	---------------	--------	----------------------

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

The MSMON CFG CSU FLT characteristics are:

Configures PARTID and PMG to measure or count in the CSU monitor selected by [MSMON_CFG_MON_SEL](#).

MSMON_CFG_CSU_FLT_s sets filter conditions for the Secure cache storage usage monitor instance selected by the Secure instance of [MSMON_CFG_MON_SEL](#). MSMON_CFG_CSU_CTL_ns sets filter conditions for the Non-secure cache storage usage monitor instance selected by the Non-secure instance of [MSMON_CFG_MON_SEL](#). MSMON_CFG_CSU_FLT_rt sets the filter conditions for the Root PARTID selected by the Root instance of [MSMON_CFG_MON_SEL](#). MSMON_CFG_CSU_FLT_rl sets the filter conditions for the Realm PARTID selected by the Realm instance of [MSMON_CFG_MON_SEL](#).

If `MPAMF_IDR.HAS_RIS` is 1, the monitor instance filter 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`.

The power domain of MSMON_CFG_CSU_FLT 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_FLT are RES0.

The power and reset domain of each MSC component is specific to that component.

MSMON_CFG_CSU_FLT is a 32-bit register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
XCLRES0		RES0PMG								PMGPARTID																PARTID							

XCL, Bits bit [31:24]
When (FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 is implemented) and MPAMF_CSUMON_IDR.HAS_XCL == 1:

Exclude Clean. The monitor instance does not count cache storage used by lines in an unmodified cache state.

XCL	Meaning
0b0	Monitor instance counts cache storage in modified and unmodified cache lines.
0b1	Monitor instance counts cache storage in modified cache lines only.

Otherwise:

Reserved, RES0.

Bits [30:24]

Reserved, RES0.

PMG, bits [23:16]

Performance monitoring group to filter cache storage usage monitoring.

If [MSMON_CFG_CSU_CTL.MATCH_PMG](#) is 0, this field is not used to match cache storage to a PMG and the contents of this field is ignored.

If [MSMON_CFG_CSU_CTL.MATCH_PMG](#) is 1 and [MSMON_CFG_CSU_CTL.MATCH_PARTID](#) is 1, the monitor instance selected by [MSMON_CFG_MON_SEL](#) measures or counts cache storage labeled with PMG equal to this field and PARTID equal to the PARTID field.

If [MSMON_CFG_CSU_CTL.MATCH_PMG](#) is 1 and [MSMON_CFG_CSU_CTL.MATCH_PARTID](#) is 0, the behavior of the monitor instance selected by [MSMON_CFG_MON_SEL](#) is CONSTRAINED UNPREDICTABLE. See [MSMON_CFG_CSU_CTL.MATCH_PMG](#) for more information.

PARTID, bits [15:0]

Partition ID to filter cache storage usage monitoring.

If [MSMON_CFG_CSU_CTL.MATCH_PARTID](#) is 0 and [MSMON_CFG_CSU_CTL.MATCH_PMG](#) is 0, the monitor measures all allocated cache storage.

If [MSMON_CFG_CSU_CTL.MATCH_PARTID](#) is 0 and [MSMON_CFG_CSU_CTL.MATCH_PMG](#) is 1, the behavior of the monitor is CONSTRAINED UNPREDICTABLE. See the description of [MSMON_CFG_CSU_CTL.MATCH_PMG](#).

If [MSMON_CFG_CSU_CTL.MATCH_PARTID](#) is 1 and [MSMON_CFG_CSU_CTL.MATCH_PMG](#) is 0, the monitor selected by [MSMON_CFG_MON_SEL](#) measures or counts cache storage labeled with PARTID equal to this field.

If [MSMON_CFG_CSU_CTL.MATCH_PARTID](#) is 1 and [MSMON_CFG_CSU_CTL.MATCH_PMG](#) is 1, the monitor selected by [MSMON_CFG_MON_SEL](#) measures or counts cache storage labeled with PARTID equal to this field and PMG equal to the PMG field.

Accessing MSMON_CFG_CSU_FLT

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:maps](#).

- [MSMON_CFG_CSU_FLT_s](#) must be accessible from the Secure MPAM feature page.
- [MSMON_CFG_CSU_FLT_ns](#) must be accessible from the Non-secure MPAM feature page.
- [MSMON_CFG_CSU_FLT_rt](#) must be accessible from the Root MPAM feature page.
- [MSMON_CFG_CSU_FLT_rl](#) must be accessible from the Realm MPAM feature page.

[MSMON_CFG_CSU_FLT_s](#), [MSMON_CFG_CSU_FLT_ns](#), [MSMON_CFG_CSU_FLT_rt](#), and [MSMON_CFG_CSU_FLT_rl](#) must be separate [registers:registers](#).

- The Secure instance ([MSMON_CFG_CSU_FLT_s](#)) accesses the PARTID and PMG matching for a cache storage usage monitor used for Secure PARTIDs.
- The Non-secure instance ([MSMON_CFG_CSU_FLT_ns](#)) accesses the PARTID and PMG matching for a cache storage usage monitor used for Non-secure PARTIDs.
- The Root instance ([MSMON_CFG_CSU_FLT_rt](#)) accesses the PARTID and PMG matching for a cache storage usage monitor used for Root PARTIDs.
- The Realm instance ([MSMON_CFG_CSU_FLT_rl](#)) accesses the PARTID and PMG matching for a cache storage usage monitor used for Realm PARTIDs.

When RIS is implemented, loads and stores to [MSMON_CFG_CSU_FLT](#) access the monitor configuration settings for the resource instance selected by [MSMON_CFG_MON_SEL](#).RIS and the cache storage usage monitor instance selected by [MSMON_CFG_MON_SEL](#).MON_SEL.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0810	MSMON_CFG_CSU_FLT_s

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0810	MSMON_CFG_CSU_FLT_ns

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0810	MSMON_CFG_CSU_FLT_rt

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x0810	MSMON_CFG_CSU_FLT_r1

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

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 Securitysecurity 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.

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](#).

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_OFLOW_CAPT 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](#).

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.

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.

Controls whether the value of [MSMON_MBWU](#) is captured on an overflow or an overflow linkage event.

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 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.

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_OFLOW_LNKG == 1:

Capture Event performs overflow behavior.

Selects whether a capture event matching the CAPT_EVNT field perform the overflow behavior or the capture behavior.

CEVNT_OFLW	Meaning
0b0	On a capture event matching the CAPT_EVNT field the capture behaviors are performed.
0b1	On a capture event matching the CAPT_EVNT field the overflow behaviors are performed.

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_FLT.PMG .

MATCH_PARTID, bit [16]

Match PARTID.

Controls whether the monitor instance counts only data transferred with PARTID matching [MSMON_CFG_MBWU_FLT.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_FLT.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_OFLOW_CAPT 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 overflow is not possible for an MBWU monitor in the MSC implementation, this field is RAZ/WI.

If the overflow interrupt is not supported by the MSC implementation, this field is RAZ/WI.

If [MPAMF_MBWUMON_IDR](#).HAS_LONG == 0, this bit is RES0.

Otherwise:

Reserved, RES0.

OFLOW_CAPT_L, Bits bit [13:8]

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 captured on an overflow or an overflow linkage event.

OFLOW_CAPT_I	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_CAPT_L, 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_CAPT_L	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:maps.**

- MSMON_CFG_MBWU_CTL_s must be accessible from the Secure MPAM feature page.
- MSMON_CFG_MBWU_CTL_ns must be accessible from the Non-secure MPAM feature page.
- MSMON_CFG_MBWU_CTL_rt must be accessible from the Root MPAM feature page.
- MSMON_CFG_MBWU_CTL_rl must 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: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 on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0828	MSMON_CFG_MBWU_CTL_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0828	MSMON_CFG_MBWU_CTL_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0828	MSMON_CFG_MBWU_CTL_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

MSMON_CFG_MBWU_FLT, MPAM Memory System Monitor Configure Memory Bandwidth Usage Monitor Filter Register

The MSMON_CFG_MBWU_FLT characteristics are:

Purpose

Controls PARTID and PMG to measure or count in the MBWU monitor selected by [MSMON_CFG_MON_SEL](#).

MSMON_CFG_MBWU_FLT_s sets filter conditions for the Secure memory bandwidth usage monitor instance selected by the Secure instance of [MSMON_CFG_MON_SEL](#). MSMON_CFG_MBWU_CTL_ns sets filter conditions for the Non-secure memory bandwidth usage monitor instance selected by the Non-secure instance of [MSMON_CFG_MON_SEL](#). MSMON_CFG_CSU_FLT_rt sets the filter conditions for the Root PARTID selected by the Root instance of [MSMON_CFG_MON_SEL](#). MSMON_CFG_CSU_FLT_rl sets the filter conditions for the Realm PARTID selected by the Realm instance of [MSMON_CFG_MON_SEL](#).

If [MPAMF_IDR.HAS_RIS](#) is 1, the monitor instance filter 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_FLT 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_FLT are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_CFG_MBWU_FLT is a 32-bit register.

Field descriptions

When FEAT_MPAMv0p1 is implemented or FEAT_MPAMv1p1 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
RWBW				RES0								PMG				PARTID															

RW filtering.

RWBW, bits [31:30]
When MPAMF_MBWUMON_IDR.HAS_RWBW == 1:

Read/write bandwidth filter. Configures the selected monitor instance to count all bandwidth, only read bandwidth or only write bandwidth.

RWBW	Meaning
0b00	Monitor instance counts read bandwidth and write bandwidth.
0b01	Monitor instance counts write bandwidth only.
0b10	Monitor instance counts read bandwidth only.
0b11	Reserved.

Otherwise:

Reserved, RES0.

Bits [29:24]

Reserved, RES0.

PMG, bits [23:16]

Performance monitoring group to filter memory bandwidth usage monitoring.

If [MSMON_CFG_MBWU_CTL.MATCH_PMG](#) == 0, this field is not used to match memory bandwidth to a PMG and the contents of this field is ignored.

If [MSMON_CFG_MBWU_CTL.MATCH_PMG](#) == 1, the monitor selected by [MSMON_CFG_MON_SEL](#) measures or counts memory bandwidth labeled with PMG equal to this field.

PARTID, bits [15:0]

Partition ID to filter memory bandwidth usage monitoring.

If [MSMON_CFG_MBWU_CTL.MATCH_PARTID](#) == 0, this field is not used to match memory bandwidth to a PARTID and the contents of this field is ignored.

If [MSMON_CFG_MBWU_CTL.MATCH_PARTID](#) == 1, the monitor selected by [MSMON_CFG_MON_SEL](#) measures or counts memory bandwidth labeled with PARTID equal to this field.

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								PMG								PARTID															

Bits [31:24]

Reserved, RES0.

PMG, bits [23:16]

Performance monitoring group to filter memory bandwidth usage monitoring.

If [MSMON_CFG_MBWU_CTL.MATCH_PMG](#) == 0, this field is not used to match memory bandwidth to a PMG and the contents of this field is ignored.

If [MSMON_CFG_MBWU_CTL.MATCH_PMG](#) == 1, the monitor selected by [MSMON_CFG_MON_SEL](#) measures or counts memory bandwidth labeled with PMG equal to this field.

PARTID, bits [15:0]

Partition ID to filter memory bandwidth usage monitoring.

If [MSMON_CFG_MBWU_CTL.MATCH_PARTID](#) == 0, this field is not used to match memory bandwidth to a PARTID and the contents of this field is ignored.

If [MSMON_CFG_MBWU_CTL.MATCH_PARTID](#) == 1, the monitor selected by [MSMON_CFG_MON_SEL](#) measures or counts memory bandwidth labeled with PARTID equal to this field.

Accessing MSMON_CFG_MBWU_FLT

This register is within the MPAM feature page memory frames.

(old)

htmldiff from-

(new)

MSMON_CFG_MON_SEL, MPAM Monitor Instance Selection Register

The MSMON_CFG_MON_SEL characteristics are:

Purpose

Selects a monitor instance to access through the MSMON configuration and counter registers.

MSMON_CFG_MON_SEL_s selects a Secure monitor instance to access via the Secure MPAM feature page.

MSMON_CFG_MON_SEL_ns selects a Non-secure monitor instance to access via the Non-secure MPAM feature page.

MSMON_CFG_MON_SEL_rt selects a Root monitor instance to access via the Root MPAM feature page.

MSMON_CFG_MON_SEL_rl selects a Realm monitor instance to access via the Realm MPAM feature page.

Note

Different performance monitoring features within an MSC could have different numbers of monitor instances. See the NUM_MON field in the corresponding ID register. This means that a monitor out-of-bounds error might be signaled when an MSMON_CFG register is accessed because the value in MSMON_CFG_MON_SEL.MON_SEL is too large for the particular monitoring feature.

To configure a monitor, set MON_SEL in this register to the index of the monitor instance to configure, then write to the MSMON_CFG_x register to set the configuration of the monitor. At a later time, read the monitor register (for example, MSMON_MBWU) to get the value of the monitor.

Configuration

The power domain of MSMON_CFG_MON_SEL is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented and (MPAMF_IDR.HAS_MSMON == 1, or (MPAMF_IDR.HAS_IMPL_IDR == 1 and MPAMF_IDR.EXT == 0) or (MPAMF_IDR.HAS_IMPL_IDR == 1, MPAMF_IDR.EXT == 1 and MPAMF_IDR.NO_IMPL_MSMON == 0)). Otherwise, direct accesses to MSMON_CFG_MON_SEL are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_CFG_MON_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								MON_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 MSMON_CFG registers.

Otherwise:

Reserved, RES0.

Bits [23:16]

Reserved, RES0.

MON_SEL, bits [15:0]

Selects the monitor instance to configure or read.

Reads and writes to other MSMON registers are indexed by MON_SEL and by the NS bit used to access MSMON_CFG_MON_SEL to access the configuration for a single monitor.

Accessing MSMON_CFG_MON_SEL

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:maps**.

- MSMON_CFG_MON_SEL_s must be accessible from the Secure MPAM feature page.
- MSMON_CFG_MON_SEL_ns must be accessible from the Non-secure MPAM feature page.
- MSMON_CFG_MON_SEL_rt must be accessible from the Root MPAM feature page.
- MSMON_CFG_MON_SEL_rl must be accessible from the Realm MPAM feature page.

MSMON_CFG_MON_SEL_s, MSMON_CFG_MON_SEL_ns, MSMON_CFG_MON_SEL_rt, and MSMON_CFG_MON_SEL_rl must be separate **registers:registers**.

- The Secure instance (MSMON_CFG_MON_SEL_s) accesses the monitor instance selector used for Secure PARTIDs.
- The Non-secure instance (MSMON_CFG_MON_SEL_ns) accesses the monitor instance selector used for Non-secure PARTIDs.
- The Root instance (MSMON_CFG_MON_SEL_rt) accesses the monitor instance selector used for Root PARTIDs.
- The Realm instance (MSMON_CFG_MON_SEL_rl) accesses the monitor instance selector used for Realm PARTIDs.

MSMON_CFG_MON_SEL can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0800	MSMON_CFG_MON_SEL_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0800	MSMON_CFG_MON_SEL_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0800	MSMON_CFG_MON_SEL_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x0800	MSMON_CFG_MON_SEL_r1

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 14:12:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MSMON_CSU, MPAM Cache Storage Usage Monitor Register

The MSMON_CSU characteristics are:

Purpose

Accesses the CSU monitor instance selected by [MSMON_CFG_MON_SEL](#).

MSMON_CSU_s is a Secure cache storage usage monitor instance selected by the Secure instance of [MSMON_CFG_MON_SEL](#). MSMON_CSU_ns is a Non-secure cache storage usage monitor instance selected by the Non-secure instance of [MSMON_CFG_MON_SEL](#). MSMON_CSU_rt is a Root cache storage usage monitor instance selected by the Root instance of [MSMON_CFG_MON_SEL](#). MSMON_CSU_rl is a Realm cache storage usage monitor instance selected by the Realm instance of [MSMON_CFG_MON_SEL](#).

If [MPAMF_IDR.HAS_RIS](#) is 1, the monitor instance 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_CSU 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_CSU are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_CSU 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
NRDY	VALUE																														

NRDY, bit [31]

Not Ready. Indicates whether the monitor instance has possibly inaccurate data.

NRDY	Meaning
0b0	The monitor instance is ready and the MSMON_CSU.VALUE field is accurate.
0b1	The monitor instance is not ready and the contents of the MSMON_CSU.VALUE field might be inaccurate or otherwise not represent the actual cache storage usage.

VALUE, bits [30:0]

Cache storage usage measurement value if MSMON_CSU.NRDY is 0. Invalid if MSMON_CSU.NRDY is 1.

VALUE is the cache storage usage measured in bytes meeting the criteria set in [MSMON_CFG_CSU_FLT](#) and [MSMON_CFG_CSU_CTL](#) for the monitor instance selected by [MSMON_CFG_MON_SEL](#).

Accessing MSMON_CSU

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:maps.~~

- MSMON_CSU_s must be accessible from the Secure MPAM feature page.
- MSMON_CSU_ns must be accessible from the Non-secure MPAM feature page.
- MSMON_CSU_rt must be accessible from the Root MPAM feature page.
- MSMON_CSU_rl must be accessible from the Realm MPAM feature page.

MSMON_CSU_s, MSMON_CSU_ns, MSMON_CSU_rt, and MSMON_CSU_rl must be separate ~~registers:registers.~~

- The Secure instance (MSMON_CSU_s) accesses the cache storage usage monitor used for Secure PARTIDs.
- The Non-secure instance (MSMON_CSU_ns) accesses the cache storage usage monitor used for Non-secure PARTIDs.
- The Root instance (MSMON_CSU_rt) accesses the cache storage usage monitor used for Root PARTIDs.
- The Realm instance (MSMON_CSU_rl) accesses the cache storage usage monitor used for Realm PARTIDs.

When RIS is implemented, reads and writes to MSMON_CSU access the cache storage usage monitor monitor instance 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, reads and writes to MSMON_CSU access the cache storage usage monitor monitor instance for the cache storage usage monitor instance selected by [MSMON_CFG_MON_SEL](#).MON_SEL.

MSMON_CSU can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0840	MSMON_CSU_s

This interface is accessible as follows:

- When MPAMF_CSUMON_IDR.CSU_RO == 0 accesses to this register are **RW**.
- When MPAMF_CSUMON_IDR.CSU_RO == 1 accesses to this register are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0840	MSMON_CSU_ns

This interface is accessible as follows:

- When MPAMF_CSUMON_IDR.CSU_RO == 0 accesses to this register are **RW**.
- When MPAMF_CSUMON_IDR.CSU_RO == 1 accesses to this register are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0840	MSMON_CSU_rt

This interface is accessible as follows:

- When FEAT_RME is implemented and MPAMF_CSUMON_IDR.CSU_RO == 0 accesses to this register are **RW**.
- When FEAT_RME is implemented and MPAMF_CSUMON_IDR.CSU_RO == 1 accesses to this register are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0840	MSMON_CSU_rl

This interface is accessible as follows:

- When FEAT_RME is implemented and MPAMF_CSUMON_IDR.CSU_RO == 0 accesses to this register are **RW**.
- When FEAT_RME is implemented and MPAMF_CSUMON_IDR.CSU_RO == 1 accesses to this register are **RO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MSMON_CSU_CAPTURE, MPAM Cache Storage Usage Monitor Capture Register

The MSMON_CSU_CAPTURE characteristics are:

Purpose

MSMON_CSU_CAPTURE is a 32-bit read-write register that accesses the captured [MSMON_CSU](#) monitor instance selected by [MSMON_CFG_MON_SEL](#).

MSMON_CSU_CAPTURE_s is the Secure cache storage usage monitor capture instance selected by the Secure instance of [MSMON_CFG_MON_SEL](#). MSMON_CSU_CAPTURE_ns is the Non-secure cache storage usage monitor capture instance selected by the Non-secure instance of [MSMON_CFG_MON_SEL](#). MSMON_CSU_CAPTURE_rt is a Root cache storage usage monitor capture instance selected by the Root instance of [MSMON_CFG_MON_SEL](#). MSMON_CSU_CAPTURE_rl is a Realm cache storage usage monitor capture instance selected by the Realm instance of [MSMON_CFG_MON_SEL](#).

If [MPAMF_IDR.HAS_RIS](#) is 1, the monitor instance capture register 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_CSU_CAPTURE 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_CAPTURE == 1. Otherwise, direct accesses to MSMON_CSU_CAPTURE are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_CSU_CAPTURE 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
NRDY	VALUE																														

NRDY, bit [31]

Not Ready. Indicates whether the captured monitor value has possibly inaccurate data.

NRDY	Meaning
0b0	The captured monitor instance was ready and the MSMON_CSU_CAPTURE.VALUE field is accurate.
0b1	The captured monitor instance was not ready and the contents of the MSMON_CSU_CAPTURE.VALUE field might be inaccurate or otherwise not represent the actual cache storage usage.

VALUE, bits [30:0]

Captured cache storage usage measurement if MSMON_CSU_CAPTURE.NRDY is 0. Invalid if MSMON_CSU_CAPTURE.NRDY is 1.

VALUE is the captured cache storage usage measurement in bytes meeting the criteria set in `MSMON_CFG_CSU_FLT` and `MSMON_CFG_CSU_CTL` for the monitor instance selected by `MSMON_CFG_MON_SEL`.

Accessing MSMON_CSU_CAPTURE

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:maps.~~

- `MSMON_CSU_CAPTURE_s` must be accessible from the Secure MPAM feature page.
- `MSMON_CSU_CAPTURE_ns` must be accessible from the Non-secure MPAM feature page.
- `MSMON_CSU_CAPTURE_rt` must be accessible from the Root MPAM feature page.
- `MSMON_CSU_CAPTURE_rl` must be accessible from the Realm MPAM feature page.

MSMON_CSU_CAPTURE_s, MSMON_CSU_CAPTURE_ns, MSMON_CSU_CAPTURE_rt, and MSMON_CSU_CAPTURE_rl must be separate registers: registers.

- The Secure instance (MSMON_CSU_CAPTURE_s) accesses the captured cache storage usage monitor used for Secure PARTIDs.
- The Non-secure instance (MSMON_CSU_CAPTURE_ns) accesses the captured cache storage usage monitor used for Non-secure PARTIDs.
- The Root instance (MSMON_CSU_CAPTURE_rt) accesses the captured cache storage usage monitor used for Root PARTIDs.
- The Realm instance (MSMON_CSU_CAPTURE_rl) accesses the captured cache storage usage monitor used for Realm PARTIDs.

When RIS is implemented, reads and writes to `MSMON_CSU_CAPTURE` access the monitor instance 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, reads and writes to `MSMON_CSU_CAPTURE` access the monitor instance for the cache storage usage monitor instance selected by `MSMON_CFG_MON_SEL.MON_SEL`.

MSMON CSU CAPTURE can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0848	MSMON_CSU_CAPTURE_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE ns	0x0848	MSMON_CSU_CAPTURE ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF BASE rt	0x0848	MSMON CSU_CAPTURE rt

When FEAT RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF BASE r1	0x0848	MSMON CSU CAPTURE r1

When FEAT RME is implemented access on this interface are **RW**.

(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 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:maps.**

- MSMON_CSU_OFSR_s must be accessible from the Secure MPAM feature page.
- MSMON_CSU_OFSR_ns must be accessible from the Non-secure MPAM feature page.
- MSMON_CSU_OFSR_rt must be accessible from the Root MPAM feature page.
- MSMON_CSU_OFSR_rl must 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: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 on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0858	MSMON_CSU_OFSR_ns

Accesses on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0858	MSMON_CSU_OFSR_rt

When FEAT_RME is implemented access on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0858	MSMON_CSU_OFSR_rl

When FEAT_RME is implemented access on this interface are **RO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ee9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MSMON_MBWU, MPAM Memory Bandwidth Usage Monitor Register

The MSMON_MBWU characteristics are:

Purpose

Accesses the monitor instance selected by [MSMON_CFG_MON_SEL](#).

MSMON_MBWU_s is the Secure memory bandwidth usage monitor instance selected by MSMON_CFG_MON_SEL_s. MSMON_MBWU_ns is the Non-secure memory bandwidth usage monitor instance selected by MSMON_CFG_MON_SEL_ns. MSMON_MBWU_rt is the Root memory bandwidth usage monitor instance selected by MSMON_CFG_MON_SEL_rt. MSMON_MBWU_rl is the Realm memory bandwidth usage monitor instance selected by MSMON_CFG_MON_SEL_rl.

If [MPAMF_IDR.HAS_RIS](#) is 1, the monitor instance register 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_MBWU 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_MBWU are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_MBWU 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
NRDY	VALUE																														

NRDY, bit [31]

Not Ready. Indicates whether the monitor has possibly inaccurate data.

NRDY	Meaning
0b0	The monitor instance is ready and the MSMON_MBWU.VALUE field is accurate.
0b1	The monitor instance is not ready and the contents of the MSMON_MBWU.VALUE field might be inaccurate or otherwise not represent the actual memory bandwidth usage.

VALUE, bits [30:0]

Memory bandwidth usage counter value if MSMON_MBWU.NRDY is 0. Invalid if MSMON_MBWU.NRDY is 1.

VALUE is the scaled count of bytes transferred since the monitor was last reset that met the criteria set in [MSMON_CFG_MBWU_FLT](#) and [MSMON_CFG_MBWU_CTL](#) for the monitor instance selected by [MSMON_CFG_MON_SEL](#).

If [MSMON_CFG_MBWU_CTL](#).SCLEN enables scaling, the count in VALUE is the number of bytes shifted right by [MPAMF_MBWUMON_IDR](#).SCALE bit positions and rounded.

Accessing MSMON_MBWU

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:maps](#).

- MSMON_MBWU_s must be accessible from the Secure MPAM feature page.
- MSMON_MBWU_ns must be accessible from the Non-secure MPAM feature page.
- MSMON_MBWU_rt must be accessible from the Root MPAM feature page.
- MSMON_MBWU_rl must be accessible from the Realm MPAM feature page.

MSMON_MBWU_s, MSMON_MBWU_ns, MSMON_MBWU_rt, and MSMON_MBWU_rl must be separate [registers:registers](#).

- The Secure instance (MSMON_MBWU_s) accesses the memory bandwidth usage monitor used for Secure PARTIDs.
- The Non-secure instance (MSMON_MBWU_ns) accesses the memory bandwidth usage monitor used for Non-secure PARTIDs.
- The Root instance (MSMON_MBWU_rt) accesses the memory bandwidth usage monitor used for Root PARTIDs.
- The Realm instance (MSMON_MBWU_rl) accesses the memory bandwidth usage monitor used for Realm PARTIDs.

When RIS is implemented, reads and writes to MSMON_MBWU access the memory bandwidth usage monitor instance for the 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, reads and writes to MSMON_MBWU access the memory bandwidth usage monitor instance for the memory bandwidth usage monitor instance selected by [MSMON_CFG_MON_SEL](#).MON_SEL.

MSMON_MBWU can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0860	MSMON_MBWU_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0860	MSMON_MBWU_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0860	MSMON_MBWU_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0860	MSMON_MBWU_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MSMON_MBWU_CAPTURE, MPAM Memory Bandwidth Usage Monitor Capture Register

The MSMON_MBWU_CAPTURE characteristics are:

Purpose

Accesses the captured MSMON_MBWU monitor instance selected by [MSMON_CFG_MON_SEL](#).

MSMON_MBWU_CAPTURE_s is the Secure memory bandwidth usage monitor capture instance selected by the Secure instance of [MSMON_CFG_MON_SEL](#). MSMON_MBWU_CAPTURE_ns is the Non-secure memory bandwidth usage monitor capture instance selected by the Non-secure instance of [MSMON_CFG_MON_SEL](#). MSMON_MBWU_CAPTURE_rt is the Root memory bandwidth usage monitor capture instance selected by the Root instance of [MSMON_CFG_MON_SEL](#). MSMON_MBWU_CAPTURE_rl is the Realm memory bandwidth usage monitor capture instance selected by the Realm instance of [MSMON_CFG_MON_SEL](#).

If [MPAMF_IDR.HAS_RIS](#) is 1, the monitor instance capture register 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_MBWU_CAPTURE 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_CAPTURE == 1. Otherwise, direct accesses to MSMON_MBWU_CAPTURE are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_MBWU_CAPTURE 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
NRDY																VALUE															

NRDY, bit [31]

Not Ready. The captured NRDY bit from the corresponding instance of [MSMON_MBWU](#). This bit indicates whether the captured monitor value has possibly inaccurate data.

NRDY	Meaning
0b0	The captured monitor instance was ready and the MSMON_MBWU_CAPTURE.VALUE field is accurate.
0b1	The captured monitor instance was not ready and the contents of the MSMON_MBWU_CAPTURE.VALUE field might be inaccurate or otherwise not represent the actual memory bandwidth usage.

VALUE, bits [30:0]

Captured memory bandwidth usage counter value if MSMON_MBWU_CAPTURE.NRDY is 0. Invalid if MSMON_MBWU_CAPTURE.NRDY is 1.

VALUE is the captured VALUE field from the corresponding instance of [MSMON_MBWU](#), the count of bytes transferred since the monitor was last reset that meet the criteria set in [MSMON_CFG_MBWU_FLT](#) and [MSMON_CFG_MBWU_CTL](#) for the monitor instance selected by [MSMON_CFG_MON_SEL](#).

VALUE captures the [MSMON_MBWU](#).VALUE and preserves any scaling that had been performed on the VALUE field in that register.

Accessing MSMON_MBWU_CAPTURE

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:maps](#).

- [MSMON_MBWU_CAPTURE_s](#) must be accessible from the Secure MPAM feature page.
- [MSMON_MBWU_CAPTURE_ns](#) must be accessible from the Non-secure MPAM feature page.
- [MSMON_MBWU_CAPTURE_rt](#) must be accessible from the Root MPAM feature page.
- [MSMON_MBWU_CAPTURE_rl](#) must be accessible from the Realm MPAM feature page.

[MSMON_MBWU_CAPTURE_s](#), [MSMON_MBWU_CAPTURE_ns](#), [MSMON_MBWU_CAPTURE_rt](#), and [MSMON_MBWU_CAPTURE_rl](#) must be separate [registers:registers](#).

- The Secure instance ([MSMON_MBWU_CAPTURE_s](#)) accesses the captured memory bandwidth usage monitor used for Secure PARTIDs.
- The Non-secure instance ([MSMON_MBWU_CAPTURE_ns](#)) accesses the captured memory bandwidth usage monitor used for Non-secure PARTIDs.
- The Root instance ([MSMON_MBWU_CAPTURE_rt](#)) accesses the captured memory bandwidth usage monitor used for Root PARTIDs.
- The Realm instance ([MSMON_MBWU_CAPTURE_rl](#)) accesses the captured memory bandwidth usage monitor used for Realm PARTIDs.

When RIS is implemented, reads and writes to [MSMON_MBWU_CAPTURE](#) access the monitor instance 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, reads and writes to [MSMON_MBWU_CAPTURE](#) access the monitor instance for the memory bandwidth usage monitor instance selected by [MSMON_CFG_MON_SEL](#).MON_SEL.

MSMON_MBWU_CAPTURE can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0868	MSMON_MBWU_CAPTURE_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0868	MSMON_MBWU_CAPTURE_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0868	MSMON_MBWU_CAPTURE_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0868	MSMON_MBWU_CAPTURE_rl

When FEAT_RME is implemented access on this interface are **RW**.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MSMON_MBWU_L, MPAM Long Memory Bandwidth Usage Monitor Register

The MSMON_MBWU_L characteristics are:

Purpose

Accesses the monitor instance selected by [MSMON_CFG_MON_SEL](#).

MSMON_MBWU_L_s is the Secure long memory bandwidth usage monitor instance selected by the Secure instance of [MSMON_CFG_MON_SEL](#). MSMON_MBWU_L_ns is the Non-secure long memory bandwidth usage monitor instance selected by the Non-secure instance of [MSMON_CFG_MON_SEL](#). MSMON_MBWU_L_rt is the Root long memory bandwidth usage monitor instance selected by MSMON_CFG_MON_SEL_rt. MSMON_MBWU_L_rl is the Realm long memory bandwidth usage monitor instance selected by MSMON_CFG_MON_SEL_rl.

If [MPAMF_IDR.HAS_RIS](#) is 1, the monitor instance long monitor register 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_MBWU_L 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_LONG == 1. Otherwise, direct accesses to MSMON_MBWU_L are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_MBWU_L is a 64-bit register.

Field descriptions

When MPAMF_MBWUMON_IDR.LWD == 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
NRDY	RES0																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

NRDY, bit [63]

Not Ready. Indicates whether the monitor instance has possibly inaccurate data.

NRDY	Meaning
0b0	The monitor instance is ready and the MSMON_MBWU_L.VALUE field is accurate.
0b1	The monitor instance is not ready and the contents of the MSMON_MBWU_L.VALUE field might be inaccurate or otherwise not represent the actual memory bandwidth usage.

Bits [62:44]

Reserved, RES0.

VALUE, bits [43:0]

Long (44-bit) memory bandwidth usage counter value if MSMON_MBWU_L.NRDY is 0. Invalid if MSMON_MBWU_L.NRDY is 1.

VALUE is the long count of bytes transferred since the monitor was last reset that met the criteria set in [MSMON_CFG_MBWU_FLT](#) and [MSMON_CFG_MBWU_CTL](#) for the monitor instance selected by [MSMON_CFG_MON_SEL](#).

When MPAMF_MBWUMON_IDR.LWD == 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	
NRDY		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	

NRDY, bit [63]

Not Ready. Indicates whether the monitor instance has possibly inaccurate data.

NRDY	Meaning
0b0	The monitor instance is ready and the MSMON_MBWU_L.VALUE field is accurate.
0b1	The monitor instance is not ready and the contents of the MSMON_MBWU_L.VALUE field might be inaccurate or otherwise not represent the actual memory bandwidth usage.

VALUE, bits [62:0]

Long (63-bit) memory bandwidth usage counter value if MSMON_MBWU_L.NRDY is 0. Invalid if MSMON_MBWU_L.NRDY is 1.

VALUE is the long count of bytes transferred since the monitor instance was last reset that met the criteria set in [MSMON_CFG_MBWU_FLT](#) and [MSMON_CFG_MBWU_CTL](#) for the monitor instance selected by [MSMON_CFG_MON_SEL](#).

Accessing MSMON_MBWU_L

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:maps](#).

- MSMON_MBWU_L_s must be accessible from the Secure MPAM feature page.
- MSMON_MBWU_L_ns must be accessible from the Non-secure MPAM feature page.
- MSMON_MBWU_L_rt must be accessible from the Root MPAM feature page.
- MSMON_MBWU_L_rl must be accessible from the Realm MPAM feature page.

MSMON_MBWU_L_s, MSMON_MBWU_L_ns, MSMON_MBWU_L_rt, and MSMON_MBWU_L_rl must be separate [registers:registers](#).

- The Secure instance (MSMON_MBWU_L_s) accesses the long memory bandwidth usage monitor used for Secure PARTIDs.
- The Non-secure instance (MSMON_MBWU_L_ns) accesses the long memory bandwidth usage monitor used for Non-secure PARTIDs.
- The Root instance (MSMON_MBWU_L_rt) accesses the long memory bandwidth usage monitor used for Root PARTIDs.
- The Realm instance (MSMON_MBWU_L_rl) accesses the long memory bandwidth usage monitor used for Realm PARTIDs.

When RIS is implemented, reads and writes to MSMON_MBWU_L access the long memory bandwidth usage monitor instance for the bandwidth resource instance selected by [MSMON_CFG_MON_SEL](#).RIS and the monitor instance selected by [MSMON_CFG_MON_SEL](#).MON_SEL.

When RIS is not implemented, reads and writes to MSMON_MBWU_L access the long memory bandwidth usage monitor instance for the monitor instance selected by [MSMON_CFG_MON_SEL](#).MON_SEL.

MSMON_MBWU_L can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0880	MSMON_MBWU_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0880	MSMON_MBWU_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0880	MSMON_MBWU_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0880	MSMON_MBWU_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbfdb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MSMON_MBWU_L_CAPTURE, MPAM Long Memory Bandwidth Usage Monitor Capture Register

The MSMON_MBWU_L_CAPTURE characteristics are:

Purpose

Accesses the captured [MSMON_MBWU_L](#) monitor instance selected by [MSMON_CFG_MON_SEL](#).

MSMON_MBWU_L_CAPTURE_s is the Secure long memory bandwidth usage monitor capture instance selected by the Secure instance of [MSMON_CFG_MON_SEL](#). MSMON_MBWU_L_CAPTURE_ns is the Non-secure long memory bandwidth usage monitor capture instance selected by the Non-secure instance of [MSMON_CFG_MON_SEL](#). MSMON_MBWU_L_CAPTURE_rt is the Root long memory bandwidth usage monitor capture instance selected by the Root instance of [MSMON_CFG_MON_SEL](#). MSMON_MBWU_L_CAPTURE_rl is the Realm long memory bandwidth usage monitor capture instance selected by the Realm instance of [MSMON_CFG_MON_SEL](#).

If [MPAMF_IDR.HAS_RIS](#) is 1, the monitor instance long capture register 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_MBWU_L_CAPTURE is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAM is implemented, MPAMF_IDR.HAS_MSMON == 1, MPAMF_MSMON_IDR.MSMON_MBWU == 1, MPAMF_MBWUMON_IDR.HAS_CAPTURE == 1 and MPAMF_MBWUMON_IDR.HAS_LONG == 1. Otherwise, direct accesses to MSMON_MBWU_L_CAPTURE are RES0.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_MBWU_L_CAPTURE is a 64-bit register.

Field descriptions

When MPAMF_MBWUMON_IDR.LWD == 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	
NRDY	RES0																			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	

NRDY, bit [63]

Not Ready. Indicates whether the monitor has possibly inaccurate data.

NRDY	Meaning
0b0	The captured monitor instance was ready and the MSMON_MBWU_L_CAPTURE.VALUE field is accurate.
0b1	The captured monitor instance was not ready and the contents of the MSMON_MBWU_L_CAPTURE.VALUE field might be inaccurate or otherwise not represent the actual memory bandwidth usage.

Bits [62:44]

Reserved, RES0.

VALUE, bits [43:0]

Captured long memory bandwidth usage counter value if MSMON_MBWU_L_CAPTURE.NRDY is 0. Invalid if MSMON_MBWU_L_CAPTURE.NRDY is 1.

VALUE is the captured 44-bit count of bytes transferred since the monitor instance was last reset that met the criteria set in [MSMON_CFG_MBWU_FLT](#) and [MSMON_CFG_MBWU_CTL](#) for the monitor instance selected by [MSMON_CFG_MON_SEL](#).

When MPAMF_MBWUMON_IDR.LWD == 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	
NRDY																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	

NRDY, bit [63]

Not Ready. Indicates whether the monitor has possibly inaccurate data.

NRDY	Meaning
0b0	The captured monitor instance was ready and the MSMON_MBWU_L_CAPTURE.VALUE field is accurate.
0b1	The captured monitor instance was not ready and the contents of the MSMON_MBWU_L_CAPTURE.VALUE field might be inaccurate or otherwise not represent the actual memory bandwidth usage.

VALUE, bits [62:0]

The captured long memory bandwidth usage counter value if MSMON_MBWU_L_CAPTURE.NRDY is 0. Invalid if MSMON_MBWU_L_CAPTURE.NRDY is 1.

VALUE is the captured 63-bit count of bytes transferred since the monitor instance was last reset that met the criteria set in [MSMON_CFG_MBWU_FLT](#) and [MSMON_CFG_MBWU_CTL](#) for the monitor instance selected by [MSMON_CFG_MON_SEL](#).

Accessing MSMON_MBWU_L_CAPTURE

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:maps](#).

- MSMON_MBWU_L_CAPTURE_s must be accessible from the Secure MPAM feature page.
- MSMON_MBWU_L_CAPTURE_ns must be accessible from the Non-secure MPAM feature page.
- MSMON_MBWU_L_CAPTURE_rt must be accessible from the Root MPAM feature page.
- MSMON_MBWU_L_CAPTURE_rl must be accessible from the Realm MPAM feature page.

MSMON_MBWU_L_CAPTURE_s, MSMON_MBWU_L_CAPTURE_ns, MSMON_MBWU_L_CAPTURE_rt, and MSMON_MBWU_L_CAPTURE_rl must be separate [registers:registers](#).

- The Secure instance (MSMON_MBWU_L_CAPTURE_s) accesses the captured long memory bandwidth usage monitor used for Secure PARTIDs.
- The Non-secure instance (MSMON_MBWU_L_CAPTURE_ns) accesses the captured long memory bandwidth usage monitor used for Non-secure PARTIDs.
- The Root instance (MSMON_MBWU_L_CAPTURE_rt) accesses the captured long memory bandwidth usage monitor used for Root PARTIDs.
- The Realm instance (MSMON_MBWU_L_CAPTURE_rl) accesses the captured long memory bandwidth usage monitor used for Realm PARTIDs.

When RIS is implemented, reads and writes to `MSMON_MBWU_L_CAPTURE` access the monitor instance 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, reads and writes to `MSMON_MBWU_L_CAPTURE` access the monitor instance for the memory bandwidth usage monitor instance selected by `MSMON_CFG_MON_SEL.MON_SEL`.

MSMON MBWU L CAPTURE can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x0890	MSMON_MBWU_CAPTURE_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF BASE ns	0x0890	MSMON MBWU CAPTURE ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE rt	0x0890	MSMON MBWU_CAPTURE rt

When FEAT RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x0890	MSMON_MBWU_CAPTURE_r1

When FEAT RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbfdb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 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 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:maps**.

- MSMON_MBWU_OFSR_s must be accessible from the Secure MPAM feature page.
- MSMON_MBWU_OFSR_ns must be accessible from the Non-secure MPAM feature page.
- MSMON_MBWU_OFSR_rt must be accessible from the Root MPAM feature page.
- MSMON_MBWU_OFSR_rl must 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: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 on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x0898	MSMON_MBWU_OFSR_ns

Accesses on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x0898	MSMON_MBWU_OFSR_rt

When FEAT_RME is implemented access on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x0898	MSMON_MBWU_OFSR_rl

When FEAT_RME is implemented access on this interface are **RO**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

MSMON_OFLOW_MSI_ADDR_H, MPAM Monitor Overflow MSI Write High-part Address Register

The MSMON_OFLOW_MSI_ADDR_H characteristics are:

Purpose

MSMON_OFLOW_MSI_ADDR_H is a 32-bit read/write register for the high part of the MPAM monitor overflow MSI address.

MSMON_OFLOW_MSI_ADDR_H_s is the high part of the MSI write address for monitor overflow interrupts from Secure monitor instances. MSMON_OFLOW_MSI_ADDR_H_ns is the high part of the MSI write address for monitor overflow interrupts from Non-secure monitor instances. MSMON_OFLOW_MSI_ADDR_H_rt is the high part of the MSI write address for monitor overflow interrupts from Root monitor instances. MSMON_OFLOW_MSI_ADDR_H_rl is the high part of the MSI write address for monitor overflow interrupts from Realm monitor instances.

Configuration

The power domain of MSMON_OFLOW_MSI_ADDR_H is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAMv1p1 is implemented and MPAMF_MSMON_IDR.HAS_OFLW_MSI == 1. Otherwise, direct accesses to MSMON_OFLOW_MSI_ADDR_H are RES0.

[MSMON_OFLOW_MSI_ADDR_L](#), [MSMON_OFLOW_MSI_ADDR_H](#), [MSMON_OFLOW_MSI_ATTR](#), [MSMON_OFLOW_MSI_DATA](#), and [MSMON_OFLOW_MSI_MPAM](#) must all be implemented to support MSI writes for monitor overflow interrupts.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_OFLOW_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 MSMON_OFLOW_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~~[maps](#).

- MSMON_OFLOW_MSI_ADDR_H_s must be accessible from the Secure MPAM feature page.
- MSMON_OFLOW_MSI_ADDR_H_ns must be accessible from the Non-secure MPAM feature page.

- MSMON_OFLW_MSI_ADDR_H_rt must be accessible from the Root MPAM feature page.
- MSMON_OFLW_MSI_ADDR_H_rl must be accessible from the Realm MPAM feature page.

MSMON_OFLW_MSI_ADDR_H_s, MSMON_OFLW_MSI_ADDR_H_ns, MSMON_OFLW_MSI_ADDR_H_rt, and MSMON_OFLW_MSI_ADDR_H_rl must be separate registers:registers.

- The Secure instance (MSMON_OFLW_MSI_ADDR_H_s) accesses the high part of the monitor overflow MSI write address of Secure monitors.
- The Non-secure instance (MSMON_OFLW_MSI_ADDR_H_ns) accesses the high part of the monitor overflow MSI write address of Non-secure monitors.
- The Root instance (MSMON_OFLW_MSI_ADDR_H_rt) accesses the high part of the monitor overflow MSI write address of Root monitors.
- The Realm instance (MSMON_OFLW_MSI_ADDR_H_rl) accesses the high part of the monitor overflow MSI write address of Realm monitors.

MSMON_OFLOW_MSI_ADDR_H can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x08E4	MSMON_OFLW_MSI_ADDR_H_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x08E4	MSMON_OFLW_MSI_ADDR_H_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x08E4	MSMON_OFLW_MSI_ADDR_H_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x08E4	MSMON_OFLW_MSI_ADDR_H_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

The MSMON_OFLOW_MSI_ADDR_L characteristics are:

Purpose

MSMON_OFLOW_MSI_ADDR_L is a 32-bit read/write register for the low part of the MPAM monitor MSI address.

MSMON_OFLOW_MSI_ADDR_L_s is the low part of the MSI write address for overflow interrupts from Secure monitor instances. MSMON_OFLOW_MSI_ADDR_L_ns is the low part of the MSI write address for overflow interrupts from Non-secure monitor instances. MSMON_OFLOW_MSI_ADDR_L_rt is the low part of the MSI write address for overflow interrupts from Root monitor instances. MSMON_OFLOW_MSI_ADDR_L_rl is the low part of the MSI write address for overflow interrupts from Realm monitor instances.

Configuration

The power domain of MSMON_OFLOW_MSI_ADDR_L is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPMV1P1 is implemented and MPAMF_MSMON_IDR.HAS_OFLW_MSI == 1. Otherwise, direct accesses to MSMON_OFLOW_MSI_ADDR_L are RES0.

`MSMON_OFLOW_MSI_ADDR_L`, `MSMON_OFLOW_MSI_ADDR_H`, `MSMON_OFLOW_MSI_ATTR`, `MSMON_OFLOW_MSI_DATA`, and `MSMON_OFLOW_MSI_MPAM` must all be implemented to support MSI writes for monitor overflow interrupts.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_OFLOW_MSI_ADDR_L 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_ADDR_L																		Bits[1:0]													

MSI_ADDR_L, bits [31:2]

MSI write address bits[31:2].

Bits [1:0]

Reads as 0b00.

Access to this field is **RO**.

Accessing MSMON_OFLOW_MSI_ADDR_L

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:maps-~~ maps.

- `MSMON_OFLOW_MSI_ADDR_L`s must be accessible from the Secure MPAM feature page.

- `MSMON_OFLOW_MSI_ADDR_L_ns` must be accessible from the Non-secure MPAM feature page.
- `MSMON_OFLOW_MSI_ADDR_L_rt` must be accessible from the Root MPAM feature page.
- `MSMON_OFLOW_MSI_ADDR_L_rl` must be accessible from the Realm MPAM feature page.

MSMON_OFLOW_MSI_ADDR_L_s, MSMON_OFLOW_MSI_ADDR_L_ns, MSMON_OFLOW_MSI_ADDR_L_rt, and MSMON_OFLOW_MSI_ADDR_L_rl must be separate registers. registers.

- The Secure instance (MSMON_OFLOW_MSI_ADDR_L_s) accesses the low part of the overflow MSI write address used for Secure PARTIDs.
- The Non-secure instance (MSMON_OFLOW_MSI_ADDR_L_ns) accesses the low part of the overflow MSI write address used for Non-secure PARTIDs.
- The Root instance (MSMON_OFLOW_MSI_ADDR_L_rt) accesses the low part of the overflow MSI write address used for Root PARTIDs.
- The Realm instance (MSMON_OFLOW_MSI_ADDR_L_rl) accesses the low part of the overflow MSI write address used for Realm PARTIDs.

MSMON_OFLOW_MSI_ADDR_L can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x08E0	MSMON_OFLOW_MSI_ADDR_L_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE ns	0x08E0	MSMON_OFLOW_MSI_ADDR_L ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x08E0	MSMON_OFLOW_MSI_ADDR_L_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ri	0x08E0	MSMON_OFLOW_MSI_ADDR_L_ri

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MSMON_OFLOW_MSI_ATTR, MPAM Monitor Overflow MSI Write Attributes Register

The MSMON_OFLOW_MSI_ATTR characteristics are:

Purpose

MSMON_OFLOW_MSI_ATTR is a 32-bit read/write register that controls MPAM monitor overflow MSI write attributes for MPAM monitor overflows in this MSC.

MSMON_OFLOW_MSI_ATTR_s controls Secure MPAM monitor overflow MSI writes. MSMON_OFLOW_MSI_ATTR_ns controls Non-secure MPAM monitor overflow MSI writes. MSMON_OFLOW_MSI_ATTR_rt controls Root MPAM monitor overflow MSI writes. MSMON_OFLOW_MSI_ATTR_rl controls Realm MPAM monitor overflow MSI writes.

Configuration

The power domain of MSMON_OFLOW_MSI_ATTR is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAMv1p1 is implemented and MPAMF_MSMON_IDR.HAS_OFLW_MSI == 1. Otherwise, direct accesses to MSMON_OFLOW_MSI_ATTR are RES0.

[MSMON_OFLOW_MSI_ADDR_L](#), [MSMON_OFLOW_MSI_ADDR_H](#), [MSMON_OFLOW_MSI_ATTR](#), [MSMON_OFLOW_MSI_DATA](#), and [MSMON_OFLOW_MSI_MPAM](#) must all be implemented to support MSI writes for monitor overflow interrupts.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_OFLOW_MSI_ATTR 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_SH				MSI_MEMATTR				RES0																		MSIEN	

Bits [31:30]

Reserved, RES0.

MSI_SH, bits [29:28]

Sharability attribute of MSI writes.

MSI_SH	Meaning
0b00	Non-shareable.
0b01	Reserved, CONSTRAINED UNPREDICTABLE.
0b10	Outer Shareable.
0b11	Inner Shareable.

When MSMON_OFLOW_MSI_ATTR.MSI_MEMATTR specifies a Device memory type, the contents of this field are IGNORED and Shareability is effectively Outer Shareable.

MSI_MEMATTR, bits [27:24]

Memory attributes of MSI writes.

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 Cachable.
0b1011	Normal Inner Write-Back Cacheable, Outer Write-Through Cachable.
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 MSMON_OFLOW_MSI_ATTR.MSI_SH are IGNORED and Shareability is effectively Outer Shareable.

Device types may be implemented as any Device type with more 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]

Monitor overflow MSI write enable.

MSIEN	Meaning
0b0	MPAM monitor overflow MSI writes are not generated to signal enabled MPAM monitor overflow interrupts. When monitor overflow MSI writes are disabled, hardwired monitor overflow interrupt could be generated if hardwired monitor overflow interrupt is implemented.
0b1	MPAM monitor overflow MSI writes are generated to signal enabled MPAM monitor overflow interrupts. When monitor overflow MSI writes are enabled, hardwired monitor overflow interrupts are not generated.

This enable affects whether a hardwired overflow interrupt is generated.

The reset behavior of this field is:

- On a MSC reset, this field resets to 0.

Accessing MSMON_OFLOW_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:maps.**

- MSMON_OFLOW_MSI_ATTR_s must be accessible from the Secure MPAM feature page.
- MSMON_OFLOW_MSI_ATTR_ns must be accessible from the Non-secure MPAM feature page.
- MSMON_OFLOW_MSI_ATTR_rt must be accessible from the Root MPAM feature page.
- MSMON_OFLOW_MSI_ATTR_rl must be accessible from the Realm MPAM feature page.

MSMON_OFLOW_MSI_ATTR_s, MSMON_OFLOW_MSI_ATTR_ns, MSMON_OFLOW_MSI_ATTR_rt, and MSMON_OFLOW_MSI_ATTR_rl must be separate **registers:registers.**

- The Secure instance (MSMON_OFLOW_MSI_ATTR_s) accesses the monitor overflow MSI write attributes of Secure monitors.
- The Non-secure instance (MSMON_OFLOW_MSI_ATTR_ns) accesses the monitor overflow MSI write attributes of Non-secure monitors.
- The Root instance (MSMON_OFLOW_MSI_ATTR_rt) accesses the monitor overflow MSI write attributes of Root monitors.
- The Realm instance (MSMON_OFLOW_MSI_ATTR_rl) accesses the monitor overflow MSI write attributes of Realm monitors.

MSMON_OFLOW_MSI_ATTR can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x08EC	MSMON_OFLOW_MSI_ATTR_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x08EC	MSMON_OFLOW_MSI_ATTR_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x08EC	MSMON_OFLOW_MSI_ATTR_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x08EC	MSMON_OFLOW_MSI_ATTR_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MSMON_OFLOW_MSI_DATA, MPAM Monitor Overflow MSI Write Data Register

The MSMON_OFLOW_MSI_DATA characteristics are:

Purpose

MSMON_OFLOW_MSI_DATA is a 32-bit read/write register for the MPAM monitor overflow MSI data.

MSMON_OFLOW_MSI_DATA_s is the data for the MSI write for monitor overflow from Secure monitor instances. MSMON_OFLOW_MSI_DATA_ns is the data for the MSI writes for monitor overflow interrupts from Non-secure monitor instances. MSMON_OFLOW_MSI_DATA_rt is the data for the MSI write for monitor overflow from Root monitor instances. MSMON_OFLOW_MSI_DATA_rl is the data for the MSI writes for monitor overflow interrupts from Realm monitor instances.

Configuration

The power domain of MSMON_OFLOW_MSI_DATA is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAMv1p1 is implemented and MPAMF_MSMON_IDR.HAS_OFLW_MSI == 1. Otherwise, direct accesses to MSMON_OFLOW_MSI_DATA are RES0.

[MSMON_OFLOW_MSI_ADDR_L](#), [MSMON_OFLOW_MSI_ADDR_H](#), [MSMON_OFLOW_MSI_ATTR](#), [MSMON_OFLOW_MSI_DATA](#), and [MSMON_OFLOW_MSI_MPAM](#) must all be implemented to support MSI writes for monitor overflow interrupts.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_OFLOW_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 write data word.

Accessing MSMON_OFLOW_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:maps.~~

- MSMON_OFLOW_MSI_DATA_s must be accessible from the Secure MPAM feature page.
- MSMON_OFLOW_MSI_DATA_ns must be accessible from the Non-secure MPAM feature page.
- MSMON_OFLOW_MSI_DATA_rt must be accessible from the Root MPAM feature page.
- MSMON_OFLOW_MSI_DATA_rl must be accessible from the Realm MPAM feature page.

MSMON_OFLOW_MSI_DATA_s, MSMON_OFLOW_MSI_DATA_ns, MSMON_OFLOW_MSI_DATA_rt, and MSMON_OFLOW_MSI_DATA_rl must be separate ~~registers:registers.~~

- The Secure instance (MSMON_OFLOW_MSI_DATA_s) accesses the monitor overflow MSI write data of Secure monitors.
- The Non-secure instance (MSMON_OFLOW_MSI_DATA_ns) accesses the monitor overflow MSI write data of Non-secure monitors.
- The Root instance (MSMON_OFLOW_MSI_DATA_rt) accesses the monitor overflow MSI write data of Root monitors.
- The Realm instance (MSMON_OFLOW_MSI_DATA_rl) accesses the monitor overflow MSI write data of Realm monitors.

MSMON_OFLOW_MSI_DATA can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_s	0x08E8	MSMON_OFLOW_MSI_DATA_s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x08E8	MSMON_OFLOW_MSI_DATA_ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x08E8	MSMON_OFLOW_MSI_DATA_rt

When FEAT_RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x08E8	MSMON_OFLOW_MSI_DATA_rl

When FEAT_RME is implemented access on this interface are **RW**.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbdbb36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

MSMON_OFLOW_MSI_MPAM, MPAM Monitor Overflow MSI Write MPAM Information Register

The MSMON_OFLOW_MSI_MPAM characteristics are:

Purpose

MSMON_OFLOW_MSI_MPAM is a 32-bit read/write register that sets the MPAM information for a monitor overflow MSI write.

MSMON_OFLOW_MSI_MPAM_s controls MPAM information labeling of Secure monitor overflow MSI writes.
 MSMON_OFLOW_MSI_MPAM_ns controls MPAM information labeling of Non-secure monitor overflow MSI writes.
 MSMON_OFLOW_MSI_MPAM_rt controls MPAM information labeling of Root monitor overflow MSI writes.
 MSMON_OFLOW_MSI_MPAM_rl controls MPAM information labeling of Realm monitor overflow MSI writes.

Configuration

The power domain of MSMON_OFLOW_MSI_MPAM is IMPLEMENTATION DEFINED.

This register is present only when FEAT_MPAMv1p1 is implemented and MPAMF_MSMON_IDR.HAS_OFLW_MSI == 1. Otherwise, direct accesses to MSMON_OFLOW_MSI_MPAM are RES0.

[MSMON_OFLOW_MSI_ADDR_L](#), [MSMON_OFLOW_MSI_ADDR_H](#), [MSMON_OFLOW_MSI_ATTR](#), [MSMON_OFLOW_MSI_DATA](#), and [MSMON_OFLOW_MSI_MPAM](#) must all be implemented to support MSI writes for monitor overflow interrupts.

The power and reset domain of each MSC component is specific to that component.

Attributes

MSMON_OFLOW_MSI_MPAM 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								PARTID															

Bits [31:24]

Reserved, RES0.

PMG, bits [23:16]

Performance monitoring group property for an MSC monitor overflow MSI write.

The reset behavior of this field is:

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

PARTID, bits [15:0]

Partition ID for an MSC monitor overflow MSI write.

The PARTID in this field is in the Secure PARTID space in the MSMON_OFLOW_MSI_MPAM_s instance and in the Non-secure PARTID space in the MSMON_OFLOW_MSI_MPAM_ns instance of this register.

The reset behavior of this field is:

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

Accessing MSMON OFLOW MSI MPAM

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:maps~~.

- `MSMON_OFLOW_MSI_MPAM_s` must be accessible from the Secure MPAM feature page.
- `MSMON_OFLOW_MSI_MPAM_ns` must be accessible from the Non-secure MPAM feature page.
- `MSMON_OFLOW_MSI_MPAM_rt` must be accessible from the Root MPAM feature page.
- `MSMON_OFLOW_MSI_MPAM_rl` must be accessible from the Realm MPAM feature page.

MSMON_OFLOW_MSI_MPAM_s, MSMON_OFLOW_MSI_MPAM_ns, MSMON_OFLOW_MSI_MPAM_rt, and MSMON_OFLOW_MSI_MPAM_rl must be separate registers. registers.

- The Secure instance (MSMON_OFLOW_MSI_MPAM_s) accesses the monitor overflow MSI MPAM information of Secure monitors.
- The Non-secure instance (MSMON_OFLOW_MSI_MPAM_ns) accesses the monitor overflow MSI MPAM information of Non-secure monitors.
- The Root instance (MSMON_OFLOW_MSI_MPAM_rt) accesses the monitor overflow MSI MPAM information of Root monitors.
- The Realm instance (MSMON_OFLOW_MSI_MPAM_rl) accesses the monitor overflow MSI MPAM information of Realm monitors.

MSMON OFLOW MSI MPAM can be accessed through the memory-mapped interfaces:

Component	Frame	Offset	Instance
MPAM	MPAMF BASE s	0x08DC	MSMON OFLOW MSI MPAM s

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE ns	0x08DC	MSMON_OFLOW MSI MPAM ns

Accesses on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE rt	0x08DC	MSMON_OFLOW_MSI_MPAM_rt

When FEAT RME is implemented access on this interface are **RW**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_r1	0x08DC	MSMON_OFLOW_MSI_MPAM_r1

When FEAT RME is implemented access on this interface are **RW**.

3020/09/2021 14:12:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd7b36e47856c443a7cc9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmlldiff 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 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:maps](#).

- MSMON_OFLOW_SR_s must be accessible from the Secure MPAM feature page.
- MSMON_OFLOW_SR_ns must be accessible from the Non-secure MPAM feature page.
- MSMON_OFLOW_SR_rt must be accessible from the Root MPAM feature page.
- MSMON_OFLOW_SR_rl must 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: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 on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_ns	0x08F0	MSMON_OFLOW_SR_ns

Accesses on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rt	0x08F0	MSMON_OFLOW_SR_rt

When FEAT_RME is implemented access on this interface are **RO**.

Component	Frame	Offset	Instance
MPAM	MPAMF_BASE_rl	0x08F0	MSMON_OFLOW_SR_rl

When FEAT_RME is implemented access on this interface are **RO**.

3020/09/2021 14:53:37; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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 **Commoncommon** architectural events and **Commoncommon** microarchitectural events are implemented, or counted, using PMU events in the range 0x0000 to 0x001F.

For more information about the **Commoncommon** 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\[31:0\]](#).

PMCEID0 is in the Core power domain.

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 **Commoncommon** event n.

For each bit:

ID<n>	Meaning
0b0	The Commoncommon event is not implemented, or not counted.
0b1	The Commoncommon event is implemented.

When the value of a bit in the field is 1, the corresponding **Commoncommon** event is implemented and counted.

Note

Arm recommends that if a **Commoncommon** 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 **Commoncommon** event.

(old)

htmldiff from-

(new)

PMCEID1, Performance Monitors Common Event Identification register 1

The PMCEID1 characteristics are:

Purpose

Defines which **Commoncommon** architectural events and **Commoncommon** microarchitectural events are implemented, or counted, using PMU events in the range 0x020 to 0x03F.

For more information about the **Commoncommon** 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\[31:0\]](#).

PMCEID1 is in the Core power domain.

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 **Commoncommon** event (0x0020 + n).

For each bit:

ID<n>	Meaning
0b0	The Commoncommon event is not implemented, or not counted.
0b1	The Commoncommon event is implemented.

When the value of a bit in the field is 1, the corresponding **Commoncommon** event is implemented and counted.

Note

Arm recommends that if a **Commoncommon** 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 **Commoncommon** event.

(old)

htmldiff from-

(new)

PMCEID2, Performance Monitors Common Event Identification register 2

The PMCEID2 characteristics are:

Purpose

Defines which **Commoncommon** architectural events and **Commoncommon** microarchitectural events are implemented, or counted, using PMU events in the range 0x4000 to 0x401F.

For more information about the **Commoncommon** 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\[31:0\]](#).

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.

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 **Commoncommon** event (0x4000 + n).

For each bit:

IDhi<n>	Meaning
0b0	The Commoncommon event is not implemented, or not counted.
0b1	The Commoncommon event is implemented.

When the value of a bit in the field is 1, the corresponding **Commoncommon** event is implemented and counted.

Note

Arm recommends that if a **Commoncommon** 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 **Commoncommon** 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 PMCEID2

Note

AllowExternalPMUAccess() has a new definition from Armv8.4. Refer to the Pseudocode definitions for more information.

PMCEID2 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xE28	PMCEID2

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.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

(old)

htmldiff from-

(new)

PMCEID3, Performance Monitors Common Event Identification register 3

The PMCEID3 characteristics are:

Purpose

Defines which **Commoncommon** architectural events and **Commoncommon** microarchitectural events are implemented, or counted, using PMU events in the range 0x4020 to 0x403F.

For more information about the **Commoncommon** 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_EL0\[63:32\]](#).

External register PMCEID3 bits [31:0] are architecturally mapped to AArch32 System register [PMCEID3\[31:0\]](#).

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.

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 **Commoncommon** event (0x4020 + n).

For each bit:

IDhi<n>	Meaning
0b0	The Commoncommon event is not implemented, or not counted.
0b1	The Commoncommon event is implemented.

When the value of a bit in the field is 1, the corresponding **Commoncommon** event is implemented and counted.

Note

Arm recommends that if a **Commoncommon** 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 **Commoncommon** 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.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856c443a7ce9a85f5e501ca

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)

PMCID2SR, CONTEXTIDR_EL2 Sample Register

The PMCID2SR characteristics are:

Purpose

Contains the sampled value of [CONTEXTIDR_EL2](#), captured on reading [PMPCSR](#)[31:0].

Configuration

PMCID2SR is in the Core power domain.

This register is present only when FEAT_PCSRv8p2 is implemented and EL2 is implemented. Otherwise, direct accesses to PMCID2SR are RES0.

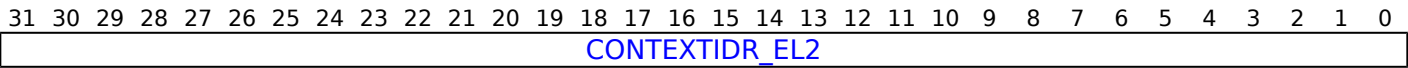
Note

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

PMCID2SR is a 32-bit register.

Field descriptions



CONTEXTIDR_EL2, bits [31:0]

Context ID. The value of [CONTEXTIDR_EL2](#) that is associated with the most recent [PMPCSR](#) sample. When the most recent [PMPCSR](#) sample ~~is~~**was** 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**~~If EL2 is using AArch32, then~~ this field is set to an UNKNOWN value.

Because the value written to PMCID2SR is an indirect read of [CONTEXTIDR_EL2](#), it is CONSTRAINED UNPREDICTABLE whether PMCID2SR is set to the original or new value if [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.

Accessing PMCID2SR

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'.

This interface is accessible as follows:

- 3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

(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\[7:0\]](#).

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 [PMCFGR](#) and the ID registers.

Attributes

PMCR_EL0 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
RAZ/WI											RES0	FZO	RES0	LP	LC	DP	X	D	C	P	E										

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.

FZO	Meaning
0b0	Do not freeze on overflow.
0b1	Event counters do not count when PMOVSCLR_EL0[(N-1):0] is nonzero, where N is the value of MDCR_EL2.HPMN if EL2 is implemented, and PMCR_EL0.N otherwise.

If EL2 is implemented, then:

In the description of this field:

- This bit affects the operation of event counters in the range [0 .. ([MDCR_EL2.HPMN](#)-1)].

If EL2 is implemented and is using AArch32, PMN is [HDCR.HPMN](#).

- If [MDCR_EL2.HPMN](#) is less than [PMCR_EL0.N](#):
 - This bit does not affect the operation of event counters in the range [[MDCR_EL2.HPMN](#) .. ([PMCR_EL0.N](#)-1)].
 - The operation of this bit ignores the values of [PMOVSCLR_EL0](#)[([PMCR_EL0.N](#)-1):[MDCR_EL2.HPMN](#)].

If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2.HPMN](#).

- This applies even when EL2 is disabled in the current Security state.

If EL2 is not implemented, PMN is [PMCR_EL0.N](#).

This bit does not affect the operation of [PMCCNTR_EL0](#).

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 implemented 0, this bit affects the operation of event counters in the range [0 .. ([PMN](#)-1)]. and [MDCR_EL2.HPMN](#) is less than [PMCR_EL0.N](#), this bit does not affect the operation of event counters in the range [[MDCR_EL2.HPMN](#):([PMCR_EL0.N](#)-1)].

The If field EL2 does is not affect the operation of other event counters implemented and .HPMN is less than [PMCR_EL0.N](#), this bit does not affect the operation of event counters in the range [[HDCR](#)[PMCCNTR_EL0](#)[HDCR](#).HPMN..([PMCR_EL0.N](#)-1)].

Note

The effect of [MDCR_EL2.HPMN](#) or [HDCR.HPMN](#) on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state. For more information, see the description of [MDCR_EL2.HPMN](#) or [HDCR.HPMN](#).

If the highest implemented Exception level is using AArch32, it is IMPLEMENTATION DEFINED whether this bit is RW or RAZ/WI.

The operation of this field applies even when EL2 is disabled in the current Security state.

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. The possible values of this bit are:

DP	Meaning
0b0	Cycle counting by PMCCNTR_EL0 is not affected by this mechanism.
0b1	When event counting is prohibited, cycle counting by PMCCNTR_EL0 is disabled in prohibited regions: <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 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. 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.

For more information, see 'Prohibiting event counting'.

When this register has an architecturally-defined reset behavior value, if this field is implemented as an RW field it resets to:

- 0 if the reset is into an Exception level that is using AArch32.
- 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.
- A value that is architecturally UNKNOWN if the reset is into an Exception level that is using AArch64.

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 PE 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, 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.

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 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. The effects of writing to this bit are:

P	Meaning
0b0	No action.
0b1	Reset all event counters, not including 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.

E	Meaning
0b0	All event counters in the range [0..(PMN-1)] and PMCCNTR_EL0 , are disabled.
0b1	All event counters in the range [0..(PMN-1)] and PMCCNTR_EL0 , are enabled by PMCNTENSET_EL0 .

If EL2 is implemented then:

In the description of this field:

- If EL2 is using AArch32, PMN is [HDCR.HPMN](#).

If EL2 is implemented and is using AArch32, PMN is [HDCR.HPMN](#).

- If EL2 is using AArch64, PMN is [MDCR_EL2.HPMN](#).

If EL2 is implemented and is using AArch64, PMN is [MDCR_EL2.HPMN](#).

- If PMN is less than PMCR_EL0.N, this bit does not affect the operation of event counters in the range [PMN..(PMCR_EL0.N-1)].

If EL2 is not implemented, PMN is PMCR_EL0.N.

If EL2 is not implemented, PMN is PMCR_EL0.N.

Note

The effect of the following fields on the operation of this bit applies if EL2 is implemented regardless of whether EL2 is enabled in the current Security state:

- [HDCR](#).HPMN. See the description of [HDCR](#).HPMN for more information.
- [MDCR_EL2](#).HPMN. See the description of [MDCR_EL2](#).HPMN for more information.

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

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:

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.

3020/09/2021 1412:5337; 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233ffbd5b36e47856c443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(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

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.

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				ARCHVERARCHID					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**.

PMEVFILTR<n>, Performance Monitors Event Type Select Register <n>, n = 0 - 30

The PMEVFILTR<n> characteristics are:

Purpose

External access to [PMEVTYPER<n>_EL0](#)[63:32].

Configuration

External register PMEVFILTR<n> bits [31:0] are architecturally mapped to AArch64 System register [PMEVTYPER<n>_EL0](#)[63:32] when AArch64 is supported.

PMEVFILTR<n> is in the Core power domain.

This register is present only when FEAT_PMUv3_TH is implemented. Otherwise, direct accesses to PMEVFILTR<n> are RES0.

Note

If FEAT_Debugv8p4 is implemented, the OPTIONAL Software Lock is not implemented.

If FEAT_DoPD is implemented, FEAT_DoubleLock is not implemented.

Attributes

PMEVFILTR<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
TC			RES0																		TH										

TC, bits [31:29]
When 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 PMEVFILTR<n>.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 PMEVFILTR<n>.TH. If PMEVFILTR<n>.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 PMEVFILTR<n>.TH.
0b010	Equals. The counter increments by V on each processor cycle when V is equal to PMEVFILTR<n>.TH.
0b011	Equals, count. The counter increments by 1 on each processor cycle when V is equal to PMEVFILTR<n>.TH.
0b100	Greater-than-or-equal. The counter increments by V on each processor cycle when V is PMEVFILTR<n>.TH or more.
0b101	Greater-than-or-equal, count. The counter increments by 1 on each processor cycle when V is PMEVFILTR<n>.TH or more.
0b110	Less-than. The counter increments by V on each processor cycle when V is less than PMEVFILTR<n>.TH.
0b111	Less-than, count. The counter increments by 1 on each processor cycle when V is less than PMEVFILTR<n>.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.

Bits [28:12]

Reserved, RES0.

TH, bits [11:0]

When FEAT_PMUv3_TH is implemented:

Threshold value. Provides the unsigned value for the threshold function defined by PMEVFILTR<n>.TC.

If PMEVFILTR<n>.TC is 0b000 and PMEVFILTR<n>.TH is zero, then the threshold function is disabled.

If [PMMIR](#).THWIDTH is less than 12, then bits PMEVFILTR<n>.TH[11:[PMMIR](#).THWIDTH] are RES0. This accounts for the behavior when writing a value greater-than-or-equal-to $2^{(\text{PMMIR.THWIDTH})}$.

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.

Accessing PMEVFILTR<n>

PMEVFILTR<n> can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0xA00 + (4 * n)	PMEVFILTR<n>

This interface is accessible as follows:

- 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**.

30/09/2021 14:53; 092b4e1bbfbb45a293b198f9330c5f529ead2b0f

Copyright © 2010-2021 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

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

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

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 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	NSUNSH	M	MTSH	T	RLK	RLU	RLH		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 in Transactional state.

T	Meaning
0b0	This bit has no effect on the filtering of events.
0b1	Do not count events in Transactional state.

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](#).

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 depends on a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.

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.
- For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the evtCount field is UNKNOWN.

Note

UNPREDICTABLE means the event must not expose privileged information.

Arm recommends that for all values that represent reserved or unsupported events, 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.

Arm recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

The reset behavior of this field is:

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

Accessing PMEVTYPER<n>_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.

PMEVTYPER<n>_EL0 can be accessed through the external debug interface:

Component	Offset	Instance
PMU	0x400 + (4 * n)	PMEVTYPER<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.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 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

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.

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								THWIDTH	BUS_WIDTH			BUS_WIDTH			BUS_SLOTS		BUS_SLOT		SLOTS		SLOTS										

Bits [31:24:20]

Reserved, RES0.

THWIDTH, bits [23:20]

PMEVFILTR<n>.TH width. Indicates implementation of the FEAT_PMuV3_TH feature, and, if implemented, the size of the **PMEVFILTR<n>.TH** field.

THWIDTH	Meaning
0b0000	FEAT_PMuV3_TH is not implemented.
0b0001	1 bit. PMEVFILTR<n>.TH [11:1] are RES0.
0b0010	2 bits. PMEVFILTR<n>.TH [11:2] are RES0.
0b0011	3 bits. PMEVFILTR<n>.TH [11:3] are RES0.
0b0100	4 bits. PMEVFILTR<n>.TH [11:4] are RES0.
0b0101	5 bits. PMEVFILTR<n>.TH [11:5] are RES0.
0b0110	6 bits. PMEVFILTR<n>.TH [11:6] are RES0.
0b0111	7 bits. PMEVFILTR<n>.TH [11:7] are RES0.
0b1000	8 bits. PMEVFILTR<n>.TH [11:8] are RES0.
0b1001	9 bits. PMEVFILTR<n>.TH [11:9] are RES0.
0b1010	10 bits. PMEVFILTR<n>.TH [11:10] are RES0.
0b1011	11 bits. PMEVFILTR<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 **PMEVFILTR<n>.TH** is $2^{(\text{PMMIR.THWIDTH})}$ minus one.

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. Defined values are:

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.

Access to this field is **RO**.

BUS_SLOTS, bits [15:8]

Bus count. The largest value by which the BUS_ACCESS event might increment **by** 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.

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** **must** **read** **not** **as** **be** 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:

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**.

3020/09/2021 1412:5337: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

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

(old)

htmldiff from-

(new)

PMPCSR, Program Counter Sample Register

The PMPCSR characteristics are:

Purpose

Holds a sampled instruction address value.

Configuration

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.

Field descriptions

63	62	61	60	59	58	57	56	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	Secure.
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 **an branch** instruction has retired since the PE left **resetReset** 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 **resetReset** state.
- No **branch** instruction has retired since the PE left **resetReset** 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], [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], [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], [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'.

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.

3020/09/2021 14:12:5237: 092b4e1bbfbb45a293b198f9330c5f529ead2b0fd4a233fbbdfb36e47856e443a7ce9a85f5e501ea

Copyright © 2010-2021 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.

(old)

htmldiff from-

(new)