

SysReg_v83A_xml-00bet4
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
SysReg_v83A_xml-00bet5

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~~ ~~Arm Limited~~ ("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~~~~ARM~~ makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party 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~~~~ARM~~'s customers is not intended to create or refer to any partnership relationship with any other company. ~~Arm~~~~ARM~~ may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any ~~click through or~~ signed written agreement ~~specifically covering~~ covering this document with ~~Arm~~~~ARM~~, then the ~~click~~~~signed~~ through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. ~~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~~~~Words~~ ~~Arm~~ corporate logo and ~~words~~~~logos~~ marked with ® or ~~TMTM~~ ~~TM~~ are registered trademarks or trademarks of ~~Arm~~~~ARM~~ Limited ~~(or its subsidiaries)~~ affiliates in the ~~US~~ ~~EU~~ and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. ~~Please~~~~You must~~ follow ~~Arm's~~~~the ARM~~ trademark usage guidelines at <http://www.arm.com/about/trademarks/guidelines/index.php>.

Copyright © 2017 ~~Arm~~~~ARM~~ Limited ~~(or its affiliates)~~~~affiliates~~. All rights reserved.

~~Arm~~~~ARM~~ Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

~~In this document, where the term ARM is used to refer to the company it means "ARM or any of its subsidiaries as appropriate".~~

~~LES-PRE-20347~~~~LES-PRE-20327~~

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg_v83A_xml-00bet4
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
SysReg_v83A_xml-00bet5

[SysReg v83A xml-00bet4](#)[\(old\)](#)htmldiff from-
SysReg v83A xml-00bet4[\(new\)](#)[SysReg v83A xml-00bet5](#)

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

AMAIRO: Auxiliary Memory Attribute Indirection Register 0

AMAIR1: Auxiliary Memory Attribute Indirection Register 1

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

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

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

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

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

DLR: Debug Link Register

DSPSR: Debug Saved Program Status Register

ELR_hyp: Exception Link Register (Hyp mode)

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

HAMAIR0: 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

HDCCR: Hyp Debug Control Register

HDFAR: Hyp Data Fault Address Register

HIFAR: Hyp Instruction Fault Address Register

HMAIR0: 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

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_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_PFR0: Processor Feature Register 0

ID_PFR1: Processor Feature Register 1

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

[PMOVSRR](#): 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

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

VMPIDR: Virtualization Multiprocessor ID Register

[VPIDR](#): Virtualization Processor ID Register

VTCR: Virtualization Translation Control Register

VTBR: Virtualization Translation Table Base Register

28/0907/2017 0816:4140

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTHP_CVAL, Counter-timer Hyp Physical CompareValue register

The CNTHP_CVAL characteristics are:

Purpose

Holds the compare value for the Hyp mode physical timer.

This register is part of:

- The Generic Timer registers functional group.
- The Virtualization registers functional group.

Configuration

AArch32 System register CNTHP_CVAL is architecturally mapped to AArch64 System register [CNTHP_CVAL_EL2](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTHP_CVAL is a 64-bit register.

Field descriptions

The CNTHP_CVAL bit assignments are:

63	62	61	60	59	58	57	56	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.

Accessing the CNTHP_CVAL

This register can be read using MRRC with the following syntax:

```
MRRC <syntax>
```

This register can be written using MCRR with the following syntax:

MCRR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	coproc	CRm
p15, 6, <Rt>, <Rt2>, c14	0110	1111	1110
p15, 2, <Rt>, <Rt2>, c14	0010	1111	1110

Accessibility

The register is accessible as follows:

<syntax>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
p15, 6, <Rt>, <Rt2>, c14	x	x	0	-	-	n/a	-
p15, 6, <Rt>, <Rt2>, c14	x	0	1	-	-	RW	RW
p15, 6, <Rt>, <Rt2>, c14	x	1	1	-	n/a	RW	RW
p15, 2, <Rt>, <Rt2>, c14	x	x	0	CNTP_CVAL	CNTP_CVAL	n/a	CNTP_CVAL
p15, 2, <Rt>, <Rt2>, c14	0	0	1	CNTP_CVAL	CNTP_CVAL	CNTP_CVAL	CNTP_CVAL
p15, 2, <Rt>, <Rt2>, c14	0	1	1	CNTP_CVAL	n/a	CNTP_CVAL	CNTP_CVAL
p15, 2, <Rt>, <Rt2>, c14	1	0	1	CNTP_CVAL	CNTP_CVAL	n/a	n/a
p15, 2, <Rt>, <Rt2>, c14	1	1	1	RW	n/a	n/a	n/a

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2.EL0PTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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.

This register is part of:

- The Generic Timer registers functional group.
- The Virtualization registers functional group.

Configuration

AArch32 System register CNTHP_TVAL is architecturally mapped to AArch64 System register [CNTHP_TVAL_EL2](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTHP_TVAL is a 32-bit register.

Field descriptions

The CNTHP_TVAL bit assignments are:

31	30	29	28	27	26	25	24	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.

Accessing the CNTHP_TVAL

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 4, <Rt>, c14, c2, 0	100	000	1110	1111	0010
p15, 0, <Rt>, c14, c2, 0	000	000	1110	1111	0010

Accessibility

The register is accessible as follows:

<syntax>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
p15, 4, <Rt>, c14, c2, 0	x	x	0	-	-	n/a	-
p15, 4, <Rt>, c14, c2, 0	x	0	1	-	-	RW	RW
p15, 4, <Rt>, c14, c2, 0	x	1	1	-	n/a	RW	RW
p15, 0, <Rt>, c14, c2, 0	x	x	0	CNTP_TVAL	CNTP_TVAL	n/a	CNTP_TVAL
p15, 0, <Rt>, c14, c2, 0	0	0	1	CNTP_TVAL	CNTP_TVAL	CNTP_TVAL	CNTP_TVAL
p15, 0, <Rt>, c14, c2, 0	0	1	1	CNTP_TVAL	n/a	CNTP_TVAL	CNTP_TVAL
p15, 0, <Rt>, c14, c2, 0	1	0	1	CNTP_TVAL	CNTP_TVAL	n/a	n/a
p15, 0, <Rt>, c14, c2, 0	1	1	1	RW	n/a	n/a	n/a

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2.EL0PTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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

The CNTHV_CVAL characteristics are:

Purpose

Provides AArch32 access to the compare value for the EL2 virtual timer.

Note

The EL2 virtual timer is implemented by ARMv8.1-VHE. It is only accessible from AArch32 state when EL0 is using AArch32, EL2 is using AArch64, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}.

This register is part of the Generic Timer registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register CNTHV_CVAL is architecturally mapped to AArch64 System register [CNTHV_CVAL_EL2](#).

This register is introduced in ARMv8.1.

Attributes

CNTHV_CVAL is a 64-bit register.

Field descriptions

The CNTHV_CVAL bit assignments are:

63	62	61	60	59	58	57	56	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](#).ENABLE is 1, the timer condition is met when ([CNTVCT](#) - 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](#).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.

Accessing the CNTHV_CVAL

This register can be read using MRRC with the following syntax:

MRRC <syntax>

This register can be written using MCRR with the following syntax:

MCRR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	coproc	CRm
p15, 3, <Rt>, <Rt2>, c14	0011	1111	1110

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	CNTV_CVAL	CNTV_CVAL	n/a	CNTV_CVAL
0	0	1	CNTV_CVAL	CNTV_CVAL	CNTV_CVAL	CNTV_CVAL
0	1	1	CNTV_CVAL	n/a	CNTV_CVAL	CNTV_CVAL
1	0	1	CNTV_CVAL	CNTV_CVAL	n/a	n/a
1	1	1	RW	n/a	n/a	n/a

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2.EL0VTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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.

Note

The EL2 virtual timer is implemented by ARMv8.1-VHE. It is only accessible from AArch32 state when EL0 is using AArch32, EL2 is using AArch64, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}.

This register is part of the Generic Timer registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register CNTHV_TVAL is architecturally mapped to AArch64 System register [CNTHV_TVAL_EL2](#).

This register is introduced in ARMv8.1.

Attributes

CNTHV_TVAL is a 32-bit register.

Field descriptions

The CNTHV_TVAL bit assignments are:

31	30	29	28	27	26	25	24	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:

- [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.

Accessing the CNTHV_TVAL

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c14, c3, 0	000	000	1110	1111	0011

This register is accessed using the encoding for [CNTV_TVAL](#).

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	CNTV_TVAL	CNTV_TVAL	n/a	CNTV_TVAL
0	0	1	CNTV_TVAL	CNTV_TVAL	CNTV_TVAL	CNTV_TVAL
0	1	1	CNTV_TVAL	n/a	CNTV_TVAL	CNTV_TVAL
1	0	1	CNTV_TVAL	CNTV_TVAL	n/a	n/a
1	1	1	RW	n/a	n/a	n/a

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2](#).EL0VTEN==0, Non-secure accesses to this register from EL0 are trapped to EL2.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTP_CVAL, Counter-timer Physical Timer CompareValue register

The CNTP_CVAL characteristics are:

Purpose

Holds the compare value for the EL1 physical timer.

This register is part of the Generic Timer registers functional group.

Configuration

AArch32 System register CNTP_CVAL is architecturally mapped to AArch64 System register [CNTP_CVAL_ELO](#).

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTP_CVAL is a 64-bit register.

Field descriptions

The CNTP_CVAL bit assignments are:

63	62	61	60	59	58	57	56	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 EL1 physical timer CompareValue.

When [CNTP_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:

- [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.

Accessing the CNTP_CVAL

This register can be read using MRRC with the following syntax:

```
MRRC <syntax>
```

This register can be written using MCRR with the following syntax:

```
MCRR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	coproc	CRm
----------	------	--------	-----

p15, 2, <Rt>, <Rt2>, c14	0010	1111	1110
--------------------------	------	------	------

Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	
EL3 using AArch32	x	x	0	RW	n/a	n/a	RW	CNTP_CVAL_s
EL3 using AArch32	x	0	1	RW	RW	RW	RW	CNTP_CVAL_ns
EL3 using AArch32	x	1	1	RW	n/a	RW	RW	CNTP_CVAL_ns
EL3 not implemented	x	x	0	RW	RW	n/a	n/a	CNTP_CVAL
EL3 not implemented	0	0	1	RW	RW	RW	n/a	CNTP_CVAL
EL3 not implemented	0	1	1	RW	n/a	RW	n/a	CNTP_CVAL
EL3 not implemented	1	0	1	RW	RW	n/a	n/a	CNTP_CVAL
EL3 not implemented	1	1	1	CNTHP_CVAL	n/a	n/a	n/a	-
EL3 using AArch64	x	x	0	RW	RW	n/a	n/a	CNTP_CVAL
EL3 using AArch64	0	0	1	RW	RW	RW	n/a	CNTP_CVAL
EL3 using AArch64	0	1	1	RW	n/a	RW	n/a	CNTP_CVAL
EL3 using AArch64	1	0	1	RW	RW	n/a	n/a	CNTP_CVAL
EL3 using AArch64	1	1	1	CNTHP_CVAL	n/a	n/a	n/a	-

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When HCR_EL2.E2H==0 :

- If [CNTKCTL_EL1](#).ELOPTEN==0, accesses to this register from EL0 are trapped to EL1.
- If [CNTKCTL](#).PLOPTEN==0, accesses to this register from EL0 are trapped to Undefined mode.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [CNTHCTL_EL2](#).EL1PCEN==0, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [CNTHCTL_EL2](#).EL1PCEN==0, and [CNTKCTL_EL1](#).ELOPTEN==1, Non-secure accesses to this register from EL0 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [CNTHCTL_EL2](#).EL1PTEN==0, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [CNTHCTL_EL2](#).EL1PTEN==0, and [CNTKCTL_EL1](#).ELOPTEN==1, Non-secure accesses to this register from EL0 are trapped to EL2.
- If [CNTKCTL_EL1](#).ELOPTEN==0, Non-secure accesses to this register from EL0 are trapped to EL1.
- If [CNTKCTL](#).PLOPTEN==0, Non-secure accesses to this register from EL0 are trapped to Undefined mode.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2](#).ELOPTEN==0, Non-secure accesses to this register from EL0 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [CNTHCTL](#).PL1PCEN==0, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
[\(old\)](#)

htmldiff from-
SysReg_v83A_xml-00bet4

[\(new\)](#)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTP_TVAL, Counter-timer Physical Timer TimerValue register

The CNTP_TVAL characteristics are:

Purpose

Holds the timer value for the EL1 physical timer.

This register is part of the Generic Timer registers functional group.

Configuration

AArch32 System register CNTP_TVAL is architecturally mapped to AArch64 System register [CNTP_TVAL_EL0](#).

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTP_TVAL is a 32-bit register.

Field descriptions

The CNTP_TVAL bit assignments are:

31	30	29	28	27	26	25	24	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.

Accessing the CNTP_TVAL

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c14, c2, 0	000	000	1110	1111	0010

Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	
EL3 not implemented	x	x	0	RW	RW	n/a	n/a	CNTP_TVAL
EL3 not implemented	0	0	1	RW	RW	RW	n/a	CNTP_TVAL
EL3 not implemented	0	1	1	RW	n/a	RW	n/a	CNTP_TVAL
EL3 not implemented	1	0	1	RW	RW	n/a	n/a	CNTP_TVAL
EL3 not implemented	1	1	1	CNTHP_TVAL	n/a	n/a	n/a	-
EL3 using AArch64	x	x	0	RW	RW	n/a	n/a	CNTP_TVAL
EL3 using AArch64	0	0	1	RW	RW	RW	n/a	CNTP_TVAL
EL3 using AArch64	0	1	1	RW	n/a	RW	n/a	CNTP_TVAL
EL3 using AArch64	1	0	1	RW	RW	n/a	n/a	CNTP_TVAL
EL3 using AArch64	1	1	1	CNTHP_TVAL	n/a	n/a	n/a	-
EL3 using AArch32	x	0	1	RW	RW	RW	RW	CNTP_TVAL_ns
EL3 using AArch32	x	1	1	RW	n/a	RW	RW	CNTP_TVAL_ns
EL3 using AArch32	x	x	0	RW	n/a	n/a	RW	CNTP_TVAL_s

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When HCR_EL2.E2H==0 :

- If [CNTKCTL_EL1](#).EL0PTEN==0, accesses to this register from EL0 are trapped to EL1.
- If [CNTKCTL](#).PL0PTEN==0, accesses to this register from EL0 are trapped to Undefined mode.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [CNTHCTL_EL2](#).EL1PCEN==0, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [CNTHCTL_EL2](#).EL1PCEN==0, and [CNTKCTL_EL1](#).EL0PTEN==1, Non-secure accesses to this register from EL0 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [CNTHCTL_EL2](#).EL1PTEN==0, Non-secure accesses to this register from EL1 are trapped to EL2.

- If [CNTHCTL_EL2](#).EL1PTEN==0, and [CNTKCTL_EL1](#).EL0PTEN==1, Non-secure accesses to this register from EL0 are trapped to EL2.
- If [CNTKCTL_EL1](#).EL0PTEN==0, Non-secure accesses to this register from EL0 are trapped to EL1.
- If [CNTKCTL](#).PL0PTEN==0, Non-secure accesses to this register from EL0 are trapped to Undefined mode.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2](#).EL0PTEN==0, Non-secure accesses to this register from EL0 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [CNTHCTL](#).PL1PCEN==0, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTV_CVAL, Counter-timer Virtual Timer CompareValue register

The CNTV_CVAL characteristics are:

Purpose

Holds the compare value for the virtual timer.

This register is part of the Generic Timer registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register CNTV_CVAL is architecturally mapped to AArch64 System register [CNTV_CVAL_EL0](#).

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTV_CVAL is a 64-bit register.

Field descriptions

The CNTV_CVAL bit assignments are:

63	62	61	60	59	58	57	56	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, bits [63:0]

Holds the EL1 virtual timer CompareValue.

When [CNTV_CTL](#).ENABLE is 1, the timer condition is met when ([CNTVCT](#) - 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:

- [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.

Accessing the CNTV_CVAL

This register can be read using MRRC with the following syntax:

```
MRRC <syntax>
```

This register can be written using MCRR with the following syntax:

```
MCRR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	coproc	CRm
----------	------	--------	-----

p15, 3, <Rt>, <Rt2>, c14	0011	1111	1110
--------------------------	------	------	------

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
0	0	1	RW	RW	RW	RW
0	1	1	RW	n/a	RW	RW
1	0	1	RW	RW	n/a	n/a
1	1	1	CNTHV_CVAL	n/a	n/a	n/a

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When HCR_EL2.E2H==0 :

- If [CNTKCTL_EL1](#).EL0VTEN==0, accesses to this register from EL0 are trapped to EL1.
- If [CNTKCTL](#).PL0VTEN==0, accesses to this register from EL0 are trapped to Undefined mode.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [CNTKCTL_EL1](#).EL0VTEN==0, Non-secure accesses to this register from EL0 are trapped to EL1.
- If [CNTKCTL](#).PL0VTEN==0, Non-secure accesses to this register from EL0 are trapped to Undefined mode.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2](#).EL0VTEN==0, Non-secure accesses to this register from EL0 are trapped to EL2.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTV_TVAL, Counter-timer Virtual Timer TimerValue register

The CNTV_TVAL characteristics are:

Purpose

Holds the timer value for the virtual timer.

This register is part of the Generic Timer registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register CNTV_TVAL is architecturally mapped to AArch64 System register [CNTV_TVAL_ELO](#).

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTV_TVAL is a 32-bit register.

Field descriptions

The CNTV_TVAL bit assignments are:

31	30	29	28	27	26	25	24	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 $(\text{CNTV_CVAL} - (-\text{CNTVOFFCNTVCTCNTPCT}))$.

On a write of this register, [CNTV_CVAL](#) is set to $(\text{CNTVCTCNTPCT} + \text{TimerValue})$, where TimerValue is treated as a signed 32-bit integer.

When [CNTP_CTL.ENABLE](#) is 1, the timer condition is met when $(\text{CNTVCTCNTPCT} - \text{CNTP_CVAL})$ 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.

Accessing the CNTV_TVAL

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c14, c3, 0	000	000	1110	1111	0011

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
0	0	1	RW	RW	RW	RW
0	1	1	RW	n/a	RW	RW
1	0	1	RW	RW	n/a	n/a
1	1	1	CNTHV_TVAL	n/a	n/a	n/a

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When HCR_EL2.E2H==0 :

- If [CNTKCTL_EL1](#).EL0VTEN==0, accesses to this register from EL0 are trapped to EL1.
- If [CNTKCTL](#).PL0VTEN==0, accesses to this register from EL0 are trapped to Undefined mode.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [CNTKCTL_EL1](#).EL0VTEN==0, Non-secure accesses to this register from EL0 are trapped to EL1.
- If [CNTKCTL](#).PL0VTEN==0, Non-secure accesses to this register from EL0 are trapped to Undefined mode.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2](#).EL0VTEN==0, Non-secure accesses to this register from EL0 are trapped to EL2.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

DBGBCR<n>, Debug Breakpoint Control Registers, n = 0 - 15

The DBGBCR<n> characteristics are:

Purpose

Holds control information for a breakpoint. Forms breakpoint n together with value register [DBGBVR<n>](#). If EL2 is implemented and this breakpoint supports Context matching, [DBGBVR<n>](#) can be associated with a Breakpoint Extended Value Register [DBGBXVR<n>](#) for VMID matching.

This register is part of the Debug registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register DBGBCR<n> is architecturally mapped to AArch64 System register [DBGBCR<n>_EL1](#).

AArch32 System register DBGBCR<n> is architecturally mapped to External register [DBGBCR<n>_EL1](#).

If breakpoint n is not implemented then this register is unallocated.

This register is in the Cold reset domain. On a Cold reset RW fields in this register reset to architecturally UNKNOWN values. The register is not affected by a Warm reset.

Attributes

DBGBCR<n> is a 32-bit register.

Field descriptions

The DBGBCR<n> bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	BT				LBN			SSC		HMC	0	0	0	0	BAS			0		0	PMC		E	

When the E field is zero, all the other fields in the register are ignored.

Bits [31:24]

Reserved, RES0.

BT, bits [23:20]

Breakpoint Type. Possible values are:

BT	Meaning
0000	Unlinked instruction address match.
0001	Linked instruction address match.
0010	Unlinked Context ID match.
0011	Linked Context ID match.
0100	Unlinked instruction address mismatch.
0101	Linked instruction address mismatch.
0110	Unlinked CONTEXTIDR_EL1 match (introduced in ARMv8.1).
0111	Linked CONTEXTIDR_EL1 match (introduced in ARMv8.1).
1000	Unlinked VMID match.
1001	Linked VMID match.
1010	Unlinked VMID and Context ID match.
1011	Linked VMID and Context ID match.
1100	Unlinked CONTEXTIDR_EL2 match (introduced in ARMv8.1).
1101	Linked CONTEXTIDR_EL2 match (introduced in ARMv8.1).
1110	Unlinked Full Context ID match (introduced in ARMv8.1).
1111	Linked Full Context ID match (introduced in ARMv8.1).

The field breaks down as follows:

- BT[3:1]: Base type.

000

Match address. [DBGBCR<n>](#) is the address of an instruction.

001

Match Context ID. [DBGBCR<n>](#).ContextID is a Context ID compared against [CONTEXTIDR](#) when ARMv8.1-VHE is not implemented, or not in a Host OS or a Host Application. When ARMv8.1-VHE is implemented, and in a Host OS or Host Application, the Context ID is compared against [CONTEXTIDR_EL2](#).

010

Mismatch address. [DBGBCR<n>](#) is the address of an instruction to be stepped.

011

Match [CONTEXTIDR_EL1](#). [DBGBCR<n>](#).ContextID is a Context ID compared against [CONTEXTIDR](#).

100

Match VMID. [DBGBCR<n>](#).VMID is a VMID compared against [VTTBR](#).VMID.

101

Match VMID and Context ID. [DBGBCR<n>](#).ContextID is a Context ID compared against [CONTEXTIDR](#), and [DBGBCR<n>](#).VMID is a VMID compared against [VTTBR](#).VMID.

110

Match [CONTEXTIDR_EL2](#). [DBGBCR<n>](#).ContextID2 is a Context ID compared against [CONTEXTIDR_EL2](#).

111

Match Full Context ID. [DBGBCR<n>](#).ContextID is compared against [CONTEXTIDR_EL1](#), and [DBGBCR<n>](#).ContextID2 is compared against [CONTEXTIDR_EL2](#).

- BT[0]: Enable linking.

Constraints on breakpoint programming mean some values are reserved under certain conditions. For more information, see 'Reserved DBGBCR<n>.BT values' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

LBN, bits [19:16]

Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to.

For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value.

This field is ignored when the value of DBGBCR<n>.E is 0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

SSC, bits [15:14]

Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see 'Reserved DBGBCR<n>.{HMC, SSC, PMC} values' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

For more information on the operation of the SSC, HMC, and PMC fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

HMC, bit [13]

Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the SSC, bits [15:14] description.

For more information on the operation of the SSC, HMC, and PMC fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bits [12:9]

Reserved, RES0.

BAS, bits [8:5]

Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state.

The permitted values depend on the breakpoint type.

For Address match breakpoints, the permitted values are:

BAS	Match instruction at	Constraint for debuggers
0011	DBGBVR<n>	Use for T32 instructions.
1100	DBGBVR<n>+2	Use for T32 instructions.
1111	DBGBVR<n>	Use for A32 instructions.

All other values are reserved. For more information, see 'Reserved DBGBCR<n>.BAS values' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

For more information on using the BAS field in Address Match breakpoints, see 'Using the BAS field in Address Match breakpoints' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

For Address mismatch breakpoints in an AArch32 stage 1 translation regime, the permitted values are:

BAS	Step instruction at	Constraint for debuggers
0000	-	Use for a match anywhere breakpoint.
0011	DBGBVR<n>	Use for stepping T32 instructions.
1100	DBGBVR<n>+2	Use for stepping T32 instructions.
1111	DBGBVR<n>	Use for stepping A32 instructions.

All other values are reserved. For more information, see 'Reserved DBGBCR<n>.BAS values' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

For more information on using the BAS field in address mismatch breakpoints, see 'Using the BAS field in Address Match breakpoints' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

For Context matching breakpoints, this field is RES1 and ignored.

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.

Bits [4:3]

Reserved, RES0.

PMC, bits [2:1]

Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>.SSC description.

For more information on the operation of the SSC, HMC, and PMC fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

E, bit [0]

Enable breakpoint [DBGBVR<n>](#). Possible values are:

E	Meaning
0	Breakpoint disabled.
1	Breakpoint enabled.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Accessing the DBGBCR<n>

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p14, 0, <Rt>, c0, <CRm>, 5	000	101	0000	1110	n<3:0>

- <CRm> is in the range c0 - c15.

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
x	0	1	-	RW	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [EDSCR](#).TDA==1, and [DBGOSLSR](#).OSLK==0, and halting is allowed, accesses to this register from PL1 and PL2 generate are trapped Software to Access Debug debug event state.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2](#).TDA==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR](#).TDA==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TDA==1, accesses to this register from EL1 and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg v83A xml-00bet4 (old)	htmldiff from-	(new)
	SysReg v83A xml-00bet4	SysReg v83A xml-00bet5

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

DBGBVR<n>, Debug Breakpoint Value Registers, n = 0 - 15

The DBGBVR<n> characteristics are:

Purpose

Holds a value for use in breakpoint matching, either the virtual address of an instruction or a context ID. Forms breakpoint n together with control register [DBGBCR<n>](#). If EL2 is implemented and this breakpoint supports Context matching, DBGBVR<n> can be associated with a Breakpoint Extended Value Register [DBGBXVR<n>](#) for VMID matching.

This register is part of the Debug registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register DBGBVR<n> is architecturally mapped to AArch64 System register [DBGBVR<n>_EL1\[31:0\]](#).

AArch32 System register DBGBVR<n> is architecturally mapped to External register [DBGBVR<n>_EL1\[31:0\]](#).

If breakpoint n is not implemented then this register is unallocated.

This register is in the Cold reset domain. On a Cold reset RW fields in this register reset to architecturally UNKNOWN values. The register is not affected by a Warm reset.

Attributes

How this register is interpreted depends on the value of [DBGBCR<n>_BT](#).

- When [DBGBCR<n>_BT](#) is 0b0x0x, this register holds a virtual address.
- When [DBGBCR<n>_BT](#) is 0b001x, 0b101x, or 0b111x, this register holds a Context ID.

For other values of [DBGBCR<n>_BT](#), this register is RES0.

Some breakpoints might not support Context ID comparison. For more information, see the description of the [DBGDIDR.CTX_CMPs](#) field.

Field descriptions

The DBGBVR<n> bit assignments are:

When DBGBCR<n>_BT==0b0x0x:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VA[31:2]																0		0													

VA[31:2], bits [31:2]

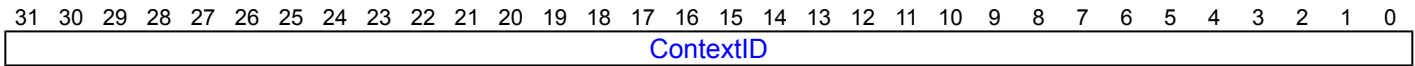
Bits[31:2] of the address value for comparison.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bits [1:0]

Reserved, RES0.

When DBGBCR<n>.BT==0b001x:



ContextID, bits [31:0]

Context ID value for comparison.

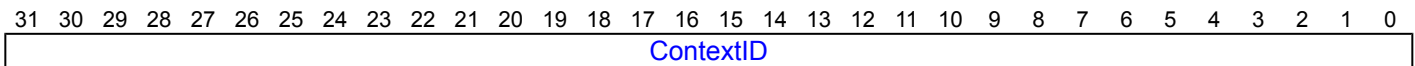
The value is compared against [CONTEXTIDR](#) in the following cases:

- The PE is in Secure state.
- EL2 is using AArch32.
- When ARMv8.1-VHE is not implemented.
- When ARMv8.1-VHE is implemented, EL2 is using AArch64, and [HCR_EL2.E2H](#) is 0.
- When ARMv8.1-VHE is implemented, EL2 is using AArch64, [HCR_EL2](#).{E2H, TGE} is {1, 0}, and the PE is in Non-secure EL0 or EL1.

When ARMv8.1-VHE is implemented, EL2 is using AArch64, [HCR_EL2](#).{E2H, TGE} is {1, 1} and the PE is in Non-secure EL0, the value is compared against [CONTEXTIDR_EL2](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

When DBGBCR<n>.BT==0b101x and EL2 implemented:

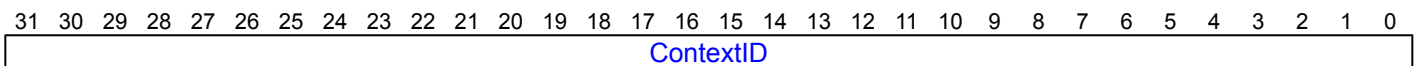


ContextID, bits [31:0]

Context ID value for comparison against [CONTEXTIDR](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

When DBGBCR<n>.BT==0b111x and EL2 implemented:



ContextID, bits [31:0]

Context ID value for comparison against [CONTEXTIDR_EL2](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Accessing the DBGBVR<n>

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p14, 0, <Rt>, c0, <CRm>, 4	000	100	0000	1110	n<3:0>

- <CRm> is in the range c0 - c15.

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
x	0	1	-	RW	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [EDSCR.TDA](#)==1, and [DBGOSLSR.OSLK](#)==0, and halting is allowed, accesses to this register from PL1 and PL2 generate are trapped Software to Access Debug debug event state.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2.TDA](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR.TDA](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3.TDA](#)==1, accesses to this register from EL1 and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

DBGXVR<n>, Debug Breakpoint Extended Value Registers, n = 0 - 15

The DBGXVR<n> characteristics are:

Purpose

Holds a value for use in breakpoint matching, to support VMID matching. Used in conjunction with a control register [DBGBCR<n>](#) and a value register [DBGBVR<n>](#), where EL2 is implemented and breakpoint n supports Context matching.

This register is part of the Debug registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register DBGXVR<n> is architecturally mapped to AArch64 System register [DBGBVR<n>_EL1\[63:32\]](#).

AArch32 System register DBGXVR<n> is architecturally mapped to External register [DBGBVR<n>_EL1\[63:32\]](#).

This register is unallocated in any of the following cases:

- Breakpoint n is not implemented.
- Breakpoint n does not support Context matching.
- EL2 is not implemented.

For more information, see the description of the [DBGDIDR.CTX_CMPs](#) field.

This register is in the Cold reset domain. On a Cold reset RW fields in this register reset to architecturally UNKNOWN values. The register is not affected by a Warm reset.

Attributes

How this register is interpreted depends on the value of [DBGBCR<n>.BT](#).

- When [DBGBCR<n>.BT](#) is 0b10xx, this register holds a VMID.
- When [DBGBCR<n>.BT](#) is 0b11xx, this register holds a Context ID.

For other values of [DBGBCR<n>.BT](#), this register is RES0.

Field descriptions

The DBGXVR<n> bit assignments are:

When DBGBCR<n>.BT==0b10xx and EL2 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VMID[15:8]				VMID[7:0]											

Bits [31:16]

Reserved, RES0.

VMID[15:8], bits [15:8]

In ARMv8.3, ARMv8.2 and ARMv8.1:

Extension to VMID[7:0]. See VMID[7:0] for more details.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.0:

Reserved, RES0.

VMID[7:0], bits [7:0]

VMID value for comparison.

The VMID is 8 bits in the following cases.

- EL2 is using AArch32.
- ARMv8.1-VMID16 is not implemented.

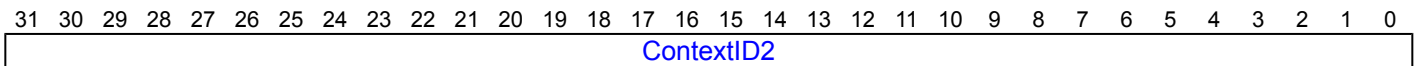
When ARMv8.1-VMID16 is implemented and EL2 is using AArch64, it is IMPLEMENTATION DEFINED whether the VMID is 8 bits or 16 bits.

VMID[15:8] is RES0 if any of the following applies:

- The implementation has an 8-bit VMID.
- [VTCR_EL2](#).VS has a value of 0.
- EL2 is using AArch32.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

When DBG BCR<n>.BT==0b11xx and EL2 implemented:



ContextID2, bits [31:0]

Context ID value for comparison against [CONTEXTIDR_EL2](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Accessing the DBG BXVR<n>

This register can be read using MRC with the following syntax:

MRC <syntax>

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p14, 0, <Rt>, c1, <CRm>, 1	000	001	0001	1110	n<3:0>

- <CRm> is in the range c0 - c15.

Accessibility

The register is accessible as follows:

Control	Accessibility
---------	---------------

E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
x	0	1	-	RW	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [EDSCR](#).TDA==1, and [DBGOSLSR](#).OSLK==0, and halting is allowed, accesses to this register from PL1 and PL2 generate a trapped Software to Access Debug debug event state.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2](#).TDA==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR](#).TDA==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TDA==1, accesses to this register from EL1 and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg_v83A_xml-00bet4 (old)	htmldiff from-	(new)
	SysReg_v83A_xml-00bet4	SysReg_v83A_xml-00bet5

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

DBGWCR<n>, Debug Watchpoint Control Registers, n = 0 - 15

The DBGWCR<n> characteristics are:

Purpose

Holds control information for a watchpoint. Forms watchpoint n together with value register [DBGWVR<n>](#).

This register is part of the Debug registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register DBGWCR<n> is architecturally mapped to AArch64 System register [DBGWCR<n>_EL1](#).

AArch32 System register DBGWCR<n> is architecturally mapped to External register [DBGWCR<n>_EL1](#).

If breakpoint n is not implemented then this register is unallocated.

This register is in the Cold reset domain. On a Cold reset RW fields in this register reset to architecturally UNKNOWN values. The register is not affected by a Warm reset.

Attributes

DBGWCR<n> is a 32-bit register.

Field descriptions

The DBGWCR<n> bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0																													

When the E field is zero, all the other fields in the register are ignored.

Bits [31:29]

Reserved, RES0.

MASK, bits [28:24]

Address mask. Only objects up to 2GB can be watched using a single mask.

MASK	Meaning
00000	No mask.
00001	Reserved.
00010	Reserved.

If programmed with a reserved value, a watchpoint must behave as if either:

- MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1.
- The watchpoint is disabled.

Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture.

Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bits [23:21]

Reserved, RES0.

WT, bit [20]

Watchpoint type. Possible values are:

WT	Meaning
0	Unlinked data address match.
1	Linked data address match.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

LBN, bits [19:16]

Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

SSC, bits [15:14]

Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.

For more information on the operation of the SSC, HMC, and PAC fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

HMC, bit [13]

Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.

For more information on the operation of the SSC, HMC, and PAC fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

BAS, bits [12:5]

Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by [DBGWVR<n>](#) is being watched.

BAS	Description
xxxxxxx1	Match byte at DBGWVR<n>
xxxxxxx1x	Match byte at DBGWVR<n>+1
xxxxx1xx	Match byte at DBGWVR<n>+2
xxxx1xxx	Match byte at DBGWVR<n>+3

In cases where [DBGWVR<n>](#) addresses a double-word:

BAS	Description, if DBGWVR<n>[2] == 0
xxx1xxxx	Match byte at DBGWVR<n>+4
xx1xxxxx	Match byte at DBGWVR<n>+5
x1xxxxxx	Match byte at DBGWVR<n>+6
1xxxxxxx	Match byte at DBGWVR<n>+7

If [DBGWVR<n>](#)[2] == 1, only BAS[3:0] are used and BAS[7:4] are ignored. ARM deprecates setting [DBGWVR<n>](#)[2] == 1.

The valid values for BAS are non-zero binary numbers all of whose set bits are contiguous. All other values are reserved and must not be used by software. See 'Reserved DBGWCR<n>.BAS values' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug)

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

LSC, bits [4:3]

Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are:

LSC	Meaning
01	Match instructions that load from a watchpointed address.
10	Match instructions that store to a watchpointed address.
11	Match instructions that load from or store to a watchpointed address.

All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

PAC, bits [2:1]

Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields.

For more information on the operation of the SSC, HMC, and PAC fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

E, bit [0]

Enable watchpoint n. Possible values are:

E	Meaning
0	Watchpoint disabled.
1	Watchpoint enabled.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Accessing the DBGWCR<n>

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p14, 0, <Rt>, c0, <CRm>, 7	000	111	0000	1110	n<3:0>

- <CRm> is in the range c0 - c15.

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3

x	x	0	-	RW	n/a	RW
x	0	1	-	RW	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [EDSCR](#).TDA==1, and [DBGOSLSR](#).OSLK==0, and halting is allowed, accesses to this register from PL1 and PL2 generate a trapped Software Access Debug event state.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2](#).TDA==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR](#).TDA==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TDA==1, accesses to this register from EL1 and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

DBGWVR<n>, Debug Watchpoint Value Registers, n = 0 - 15

The DBGWVR<n> characteristics are:

Purpose

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register [DBGWCR<n>](#).

This register is part of the Debug registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register DBGWVR<n> is architecturally mapped to AArch64 System register [DBGWVR<n>_EL1\[31:0\]](#).

AArch32 System register DBGWVR<n> is architecturally mapped to External register [DBGWVR<n>_EL1\[31:0\]](#).

If breakpoint n is not implemented then this register is unallocated.

This register is in the Cold reset domain. On a Cold reset RW fields in this register reset to architecturally UNKNOWN values. The register is not affected by a Warm reset.

Attributes

DBGWVR<n> is a 32-bit register.

Field descriptions

The DBGWVR<n> bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VA																0		0													

VA, bits [31:2]

Bits[31:2] of the address value for comparison.

ARM deprecates setting [DBGWVR<n>\[2\]](#) == 1.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bits [1:0]

Reserved, RES0.

Accessing the DBGWVR<n>

This register can be read using MRC with the following syntax:

MRC <syntax>

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p14, 0, <Rt>, c0, <CRm>, 6	000	110	0000	1110	n<3:0>

- <CRm> is in the range c0 - c15.

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
x	0	1	-	RW	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [EDSCR.TDA](#)==1, and [DBGOSLSR.OSLK](#)==0, and halting is allowed, accesses to this register from PL1 and PL2 generate are trapped Software to Access Debug debug event state.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2.TDA](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3.TDA](#)==1, accesses to this register from EL1 and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

ICC_IGRPEN0, Interrupt Controller Interrupt Group 0 Enable register

The ICC_IGRPEN0 characteristics are:

Purpose

Controls whether Group 0 interrupts are enabled or not.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register ICC_IGRPEN0 is architecturally mapped to AArch64 System register [ICC_IGRPEN0_EL1](#).

Attributes

ICC_IGRPEN0 is a 32-bit register.

Field descriptions

The ICC_IGRPEN0 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Enable

Bits [31:1]

Reserved, RES0.

Enable, bit [0]

Enables Group 0 interrupts.

Enable	Meaning
0	Group 0 interrupts are disabled.
1	Group 0 interrupts are enabled.

Virtual accesses to this register update [ICH_VMCR](#).VENG0.

When this register has an architecturally-defined reset value, this field resets to 0.

Accessing the ICC_IGRPEN0

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c12, c12, 6	000	110	1100	1111	1100
p15, 0, <Rt>, c12, c12, 6	000	110	1100	1111	1100

When HCR.FMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV_IGRPEN0](#).

Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	RW	n/a	RW
x	x	1	1	-	n/a	RW	RW
0	x	0	1	-	RW	RW	RW
1	x	0	1	-	ICV_IGRPEN0	RW	RW

<syntax>	Control				Accessibility			
	FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
p15, 0, <Rt>, c12, c12, 6	x	x	x	0	-	RW	n/a	RW
p15, 0, <Rt>, c12, c12, 6	x	x	1	1	-	n/a	RW	RW
p15, 0, <Rt>, c12, c12, 6	0	x	0	1	-	RW	RW	RW
p15, 0, <Rt>, c12, c12, 6	1	x	0	1	-	ICV_IGRPEN0	RW	RW

This table applies to all instructions that can access this register.

ICC_IGRPEN0 is only accessible at Non-secure EL1 when HCR.FMO is set to 0.

Note

When HCR.FMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC_IGRPEN0 results in an access to [ICV_IGRPEN0](#).

The lowest Exception level at which this register can be accessed is governed by the Exception level to which FIQ is routed. This routing depends on SCR.FIQ, SCR.NS and HCR.FMO.

If an interrupt is pending within the CPU interface when Enable becomes 0, the interrupt must be released to allow the Distributor to forward the interrupt to a different PE.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [ICC_SRE](#).SRE==0, accesses to this register from EL1 are UNDEFINED.
- If [ICC_HSRE](#).SRE==0, accesses to this register from EL2 are UNDEFINED.
- If [ICC_MSRE](#).SRE==0, accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [HSTR_EL2](#).T12==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2](#).T12==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HSTR](#).T12==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When SCR_EL3.NS==1 :

- If [ICH_HCR](#).TALL0==1, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [ICH_HCR_EL2](#).TALL0==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch32 :

- If [SCR](#).FIQ==1, and EL3 is implemented and configured to use AArch32, accesses to this register from EL2 and EL3 modes other than Monitor mode are UNDEFINED.

When EL3 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [SCR](#).FIQ==1, and [HCR](#).FMO==0, and EL2 is implemented and configured to use AArch32, Non-secure accesses to this register from EL1 are UNDEFINED.

When EL3 is implemented and is using AArch64 and SCR_EL3.NS==0 :

- If [SCR_EL3](#).FIQ==1, and EL3 is implemented and configured to use AArch64, Secure accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR_EL3](#).FIQ==1, and EL3 is implemented and configured to use AArch64, accesses to this register from EL2 are trapped to EL3.

When EL3 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [SCR_EL3](#).FIQ==1, and [HCR](#).FMO==0, and EL3 is implemented and configured to use AArch64 and EL2 is implemented and configured to use AArch32, Non-secure accesses to this register from EL1 are trapped to EL3.
- If [SCR_EL3](#).FIQ==1, and [HCR_EL2](#).FMO==0, and EL2 is implemented and configured to use AArch64, Non-secure accesses to this register from EL1 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

ICC_IGRPEN1, Interrupt Controller Interrupt Group 1 Enable register

The ICC_IGRPEN1 characteristics are:

Purpose

Controls whether Group 1 interrupts are enabled for the current Security state.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

Configuration

AArch32 System register ICC_IGRPEN1 (S) is architecturally mapped to AArch64 System register [ICC_IGRPEN1_EL1 \(S\)](#).

AArch32 System register ICC_IGRPEN1 (NS) is architecturally mapped to AArch64 System register [ICC_IGRPEN1_EL1 \(NS\)](#).

Attributes

ICC_IGRPEN1 is a 32-bit register.

Field descriptions

The ICC_IGRPEN1 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Enable

Bits [31:1]

Reserved, RES0.

Enable, bit [0]

Enables Group 1 interrupts for the current Security state.

Enable	Meaning
0	Group 1 interrupts are disabled for the current Security state.
1	Group 1 interrupts are enabled for the current Security state.

Virtual accesses to this register update [ICH_VMCR.VENG1](#).

If EL3 is present:

- This bit is a read/write alias of [ICC_MGRPEN1.EnableGrp1 {S, NS}](#) as appropriate if EL3 is using AArch32, or [ICC_IGRPEN1_EL3.EnableGrp1 {S, NS}](#) as appropriate if EL3 is using AArch64.
- When this register is accessed at EL3, the copy of this register appropriate to the current setting of SCR.NS is accessed.

When this register has an architecturally-defined reset value, this field resets to 0.

Accessing the ICC_IGRPEN1

This register can be read using MRC with the following syntax:

MRC <syntax>

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c12, c12, 7	000	111	1100	1111	1100
p15, 0, <Rt>, c12, c12, 7	000	111	1100	1111	1100

When HCR.IMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV_IGRPEN1](#).

Accessibility

The register is accessible as follows:

Configuration	Control				Accessibility				Instance
	FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3	
EL3 not implemented	x	x	x	0	-	RW	n/a	n/a	ICC_IGRPEN1
EL3 not implemented	x	x	1	1	-	n/a	RW	n/a	ICC_IGRPEN1
EL3 not implemented	x	0	0	1	-	RW	RW	n/a	ICC_IGRPEN1
EL3 not implemented	x	1	0	1	-	ICV_IGRPEN1	RW	n/a	ICC_IGRPEN1
EL3 using AArch64	x	x	1	1	-	n/a	RW	n/a	ICC_IGRPEN1_ns
EL3 using AArch64	x	0	0	1	-	RW	RW	n/a	ICC_IGRPEN1_ns
EL3 using AArch64	x	1	0	1	-	ICV_IGRPEN1	RW	n/a	ICC_IGRPEN1_ns
EL3 using AArch32	x	x	1	1	-	n/a	RW	RW	ICC_IGRPEN1_ns
EL3 using AArch32	x	0	0	1	-	RW	RW	RW	ICC_IGRPEN1_ns
EL3 using AArch32	x	1	0	1	-	ICV_IGRPEN1	RW	RW	ICC_IGRPEN1_ns
EL3 using AArch64	x	x	x	0	-	RW	n/a	n/a	ICC_IGRPEN1_s
EL3 using AArch32	x	x	x	0	-	-	-	RW	ICC_IGRPEN1_s

<syntax>	Configuration	Control				Accessibility				Instance
		FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3	
p15, 0, <Rt>, c12, c12, 7	EL3 not implemented	x	x	x	0	-	RW	n/a	n/a	ICC_IGRPEN1
p15, 0, <Rt>, c12, c12, 7	EL3 not implemented	x	x	1	1	-	n/a	RW	n/a	ICC_IGRPEN1
p15, 0, <Rt>, c12, c12, 7	EL3 not implemented	x	0	0	1	-	RW	RW	n/a	ICC_IGRPEN1
p15, 0, <Rt>, c12, c12, 7	EL3 not implemented	x	1	0	1	-	ICV_IGRPEN1	RW	n/a	ICC_IGRPEN1
p15, 0, <Rt>, c12, c12, 7	EL3 using AArch64	x	x	1	1	-	n/a	RW	n/a	ICC_IGRPEN1_ns
p15, 0, <Rt>, c12, c12, 7	EL3 using AArch64	x	0	0	1	-	RW	RW	n/a	ICC_IGRPEN1_ns
p15, 0, <Rt>, c12, c12, 7	EL3 using AArch64	x	1	0	1	-	ICV_IGRPEN1	RW	n/a	ICC_IGRPEN1_ns
p15, 0, <Rt>, c12, c12, 7	EL3 using AArch32	x	x	1	1	-	n/a	RW	RW	ICC_IGRPEN1_ns
p15, 0, <Rt>, c12, c12, 7	EL3 using AArch32	x	0	0	1	-	RW	RW	RW	ICC_IGRPEN1_ns

p15, 0, <Rt>, e12, e12, 7	EL3 using AArch32	x	1	0	1	-	ICV_IGRPEN1	RW	RW	ICC_IGRPEN1_ns
p15, 0, <Rt>, e12, e12, 7	EL3 using AArch64	x	x	x	0	-	RW	n/a	n/a	ICC_IGRPEN1_s
p15, 0, <Rt>, e12, e12, 7	EL3 using AArch32	x	x	x	0	-	-	-	RW	ICC_IGRPEN1_s

This table applies to all instructions that can access this register.

ICC_IGRPEN1 is only accessible at Non-secure EL1 when HCR.IMO is set to 0.

Note

When HCR.IMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC_IGRPEN1 results in an access to [ICV_IGRPEN1](#).

The lowest Exception level at which this register can be accessed is governed by the Exception level to which IRQ is routed. This routing depends on SCR.IRQ, SCR.NS and HCR.IMO.

If an interrupt is pending within the CPU interface when Enable becomes 0, the interrupt must be released to allow the Distributor to forward the interrupt to a different PE.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [ICC_SRE](#).SRE==0, accesses to this register from EL1 are UNDEFINED.
- If [ICC_HSRE](#).SRE==0, accesses to this register from EL2 are UNDEFINED.
- If [ICC_MSRE](#).SRE==0, accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [HSTR_EL2](#).T12==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2](#).T12==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HSTR](#).T12==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When SCR_EL3.NS==1 :

- If [ICH_HCR](#).TALL1==1, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [ICH_HCR_EL2](#).TALL1==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch32 :

- If [SCR](#).IRQ==1, accesses to this register from EL2 and EL3 modes other than Monitor mode are UNDEFINED.

When EL3 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [SCR](#).IRQ==1, and [HCR](#).IMO==0, Non-secure accesses to this register from EL1 are UNDEFINED.

When EL3 is implemented and is using AArch64 and SCR_EL3.NS==0 :

- If [SCR_EL3](#).IRQ==1, Secure accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR_EL3](#).IRQ==1, accesses to this register from EL2 are trapped to EL3.

When EL3 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [SCR_EL3](#).IRQ==1, and [HCR](#).IMO==0, Non-secure accesses to this register from EL1 are trapped to EL3.
- If [SCR_EL3](#).IRQ==1, and [HCR_EL2](#).IMO==0, Non-secure accesses to this register from EL1 are trapped to EL3.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

ICV_IGRPEN0, Interrupt Controller Virtual Interrupt Group 0 Enable register

The ICV_IGRPEN0 characteristics are:

Purpose

Controls whether virtual Group 0 interrupts are enabled or not.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

Configuration

AArch32 System register ICV_IGRPEN0 is architecturally mapped to AArch64 System register [ICV_IGRPEN0_EL1](#).

Attributes

ICV_IGRPEN0 is a 32-bit register.

Field descriptions

The ICV_IGRPEN0 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Enable

Bits [31:1]

Reserved, RES0.

Enable, bit [0]

Enables virtual Group 0 interrupts.

Enable	Meaning
0	Virtual Group 0 interrupts are disabled.
1	Virtual Group 0 interrupts are enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

Accessing the ICV_IGRPEN0

This register can be read using MRC with the following syntax:

MRC <syntax>

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c12, c12, 6	000	110	1100	1111	1100
p15, 0, <Rt>, c12, c12, 6	000	110	1100	1111	1100

When HCR.FMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC_IGRPEN0](#).

Accessibility

The register is accessible as follows:

Control				Accessibility				
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3	
x	x	x	0	-	ICC_IGRPEN0	n/a	ICC_IGRPEN0	
x	x	1	1	-	n/a	ICC_IGRPEN0	ICC_IGRPEN0	
0	x	0	1	-	ICC_IGRPEN0	ICC_IGRPEN0	ICC_IGRPEN0	
1	x	0	1	-	RW	ICC_IGRPEN0	ICC_IGRPEN0	
<syntax>				Accessibility				
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3	
p15, 0, <Rt>, c12, c12, 6	x	x	x	0	-	ICC_IGRPEN0	n/a	ICC_IGRPEN0
p15, 0, <Rt>, c12, c12, 6	x	x	1	1	-	n/a	ICC_IGRPEN0	ICC_IGRPEN0
p15, 0, <Rt>, c12, c12, 6	0	x	0	1	-	ICC_IGRPEN0	ICC_IGRPEN0	ICC_IGRPEN0
p15, 0, <Rt>, c12, c12, 6	1	x	0	1	-	RW	ICC_IGRPEN0	ICC_IGRPEN0

This table applies to all instructions that can access this register.

ICV_IGRPEN0 is only accessible at Non-secure EL1 when HCR.FMO is set to 1.

Note

When HCR.FMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV_IGRPEN0 results in an access to [ICC_IGRPEN0](#).

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [ICC_SRE](#).SRE==0, Non-secure accesses to this register from EL1 are UNDEFINED.
- If [ICC_SRE_EL1](#).SRE==0, Non-secure accesses to this register from EL1 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [HSTR_EL2](#).T12==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2](#).T12==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HSTR](#).T12==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When SCR_EL3.NS==1 :

- If [ICH_HCR](#).TALL0==1, Non-secure accesses to this register from EL1 are trapped to EL2.

- If [ICH_HCR_EL2](#).TALL0==1, Non-secure accesses to this register from EL1 are trapped to EL2.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg_v83A_xml-00bet4	htmldiff from-	(new)
(old)	SysReg_v83A_xml-00bet4	SysReg_v83A_xml-00bet5

ICV_IGRPEN1, Interrupt Controller Virtual Interrupt Group 1 Enable register

The ICV_IGRPEN1 characteristics are:

Purpose

Controls whether virtual Group 1 interrupts are enabled for the current Security state.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

Configuration

AArch32 System register ICV_IGRPEN1 is architecturally mapped to AArch64 System register [ICV_IGRPEN1_EL1](#).

Attributes

ICV_IGRPEN1 is a 32-bit register.

Field descriptions

The ICV_IGRPEN1 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Enable

Bits [31:1]

Reserved, RES0.

Enable, bit [0]

Enables virtual Group 1 interrupts.

Enable	Meaning
0	Virtual Group 1 interrupts are disabled.
1	Virtual Group 1 interrupts are enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

Accessing the ICV_IGRPEN1

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c12, c12, 7	000	111	1100	1111	1100
p15, 0, <Rt>, c12, c12, 7	000	111	1100	1111	1100

When HCR.IMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC_IGRPEN1](#).

Accessibility

The register is accessible as follows:

Control				Accessibility				
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3	
x	x	x	0	-	ICC_IGRPEN1	n/a	ICC_IGRPEN1	
x	x	1	1	-	n/a	ICC_IGRPEN1	ICC_IGRPEN1	
x	0	0	1	-	ICC_IGRPEN1	ICC_IGRPEN1	ICC_IGRPEN1	
x	1	0	1	-	RW	ICC_IGRPEN1	ICC_IGRPEN1	
<syntax>				Accessibility				
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3	
p15, 0, <Rt>, c12, c12, 7	x	x	0	-	ICC_IGRPEN1	n/a	ICC_IGRPEN1	
p15, 0, <Rt>, c12, c12, 7	x	x	1	-	n/a	ICC_IGRPEN1	ICC_IGRPEN1	
p15, 0, <Rt>, c12, c12, 7	x	0	1	-	ICC_IGRPEN1	ICC_IGRPEN1	ICC_IGRPEN1	
p15, 0, <Rt>, c12, c12, 7	x	1	1	-	RW	ICC_IGRPEN1	ICC_IGRPEN1	

This table applies to all instructions that can access this register.

ICV_IGRPEN1 is only accessible at Non-secure EL1 when HCR.IMO is set to 1.

Note

When HCR.IMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV_IGRPEN1 results in an access to [ICC_IGRPEN1](#).

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [ICC_SRE](#).SRE==0, Non-secure accesses to this register from EL1 are UNDEFINED.
- If [ICC_SRE_EL1](#).SRE==0, Non-secure accesses to this register from EL1 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [HSTR_EL2](#).T12==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2](#).T12==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HSTR](#).T12==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When SCR_EL3.NS==1 :

- If [ICH_HCR](#).TALL1==1, Non-secure accesses to this register from EL1 are trapped to EL2.

- If [ICH_HCR_EL2](#).TALL1==1, Non-secure accesses to this register from EL1 are trapped to EL2.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg_v83A_xml-00bet4 (old)	htmldiff from-	(new)
	SysReg_v83A_xml-00bet4	SysReg_v83A_xml-00bet5

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

ID_MMFR3, Memory Model Feature Register 3

The ID_MMFR3 characteristics are:

Purpose

Provides information about the implemented memory model and memory management support in AArch32 state.

Must be interpreted with [ID_MMFR0](#), [ID_MMFR1](#), [ID_MMFR2](#), and [ID_MMFR4](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section G4.14.6.

This register is part of the Identification registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register ID_MMFR3 is architecturally mapped to AArch64 System register [ID_MMFR3_EL1](#).

Attributes

ID_MMFR3 is a 32-bit register.

Field descriptions

The ID_MMFR3 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Supersec				CMemSz				CohWalk				PAN				MaintBcst				BPMaint				CMaintSW				CMaintVA			

Supersec, bits [31:28]

Supersections. On a VMSA implementation, indicates whether Supersections are supported. Defined values are:

Supersec	Meaning
0000	Supersections supported.
1111	Supersections not supported.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 1111.

CMemSz, bits [27:24]

Cached Memory Size. Indicates the physical memory size supported by the caches. Defined values are:

CMemSz	Meaning
0000	4GB, corresponding to a 32-bit physical address range.
0001	64GB, corresponding to a 36-bit physical address range.
0010	1TB or more, corresponding to a 40-bit or larger physical address range.

All other values are reserved.

In ARMv8-A the permitted values are 0000, 0001, and 0010.

CohWalk, bits [23:20]

Coherent Walk. Indicates whether Translation table updates require a clean to the point of unification. Defined values are:

CohWalk	Meaning
0000	Updates to the translation tables require a clean to the point of unification to ensure visibility by subsequent translation table walks.
0001	Updates to the translation tables do not require a clean to the point of unification to ensure visibility by subsequent translation table walks.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

PAN, bits [19:16]

In ARMv8.3, ARMv8.2 and ARMv8.1:

Privileged Access Never. Indicates support for the PAN bit in [CPSR](#), [SPSR](#), and [DSPSR](#) in AArch32 state. Defined values are:

PAN	Meaning
0000	PAN not supported.
0001	PAN supported.
0010	PAN supported and ATS1CPRP and ATS1CPWP instructions supported.

All other values are reserved.

ARMv8.1-PAN implements the functionality identified by the value 0001.

ARMv8.2-ATS1E1 implements the functionality added by the value 0010.

In ARMv8.1 the value ~~is~~ 0000 is not permitted.

From ARMv8.2, the only permitted value is 0010.

In ARMv8.0:

Reserved, RES0.

MaintBcst, bits [15:12]

Maintenance Broadcast. Indicates whether Cache, TLB, and branch predictor operations are broadcast. Defined values are:

MaintBcst	Meaning
0000	Cache, TLB, and branch predictor operations only affect local structures.
0001	Cache and branch predictor operations affect structures according to shareability and defined behavior of instructions. TLB operations only affect local structures.
0010	Cache, TLB, and branch predictor operations affect structures according to shareability and defined behavior of instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

BPMaint, bits [11:8]

Branch Predictor Maintenance. Indicates the supported branch predictor maintenance operations in an implementation with hierarchical cache maintenance operations. Defined values are:

BPMaint	Meaning
0000	None supported.
0001	Supported branch predictor maintenance operations are: <ul style="list-style-type: none"> • Invalidate all branch predictors.
0010	As for 0001, and adds: <ul style="list-style-type: none"> • Invalidate branch predictors by VA.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

CMaintSW, bits [7:4]

Cache Maintenance by Set/Way. Indicates the supported cache maintenance operations by set/way, in an implementation with hierarchical caches. Defined values are:

CMaintSW	Meaning
0000	None supported.
0001	Supported hierarchical cache maintenance instructions by set/way are: <ul style="list-style-type: none"> • Invalidate data cache by set/way. • Clean data cache by set/way. • Clean and invalidate data cache by set/way.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

In a unified cache implementation, the data cache maintenance operations apply to the unified caches.

CMaintVA, bits [3:0]

Cache Maintenance by Virtual Address. Indicates the supported cache maintenance operations by VA, in an implementation with hierarchical caches. Defined values are:

CMaintVA	Meaning
0000	None supported.
0001	Supported hierarchical cache maintenance operations by VA are: <ul style="list-style-type: none"> • Invalidate data cache by VA. • Clean data cache by VA. • Clean and invalidate data cache by VA. • Invalidate instruction cache by VA. • Invalidate all instruction cache entries.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

In a unified cache implementation, data cache maintenance operations apply to the unified caches, and the instruction cache maintenance instructions are not implemented.

Accessing the ID_MMFR3

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c0, c1, 7	000	111	0000	1111	0001

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RO	n/a	RO
x	0	1	-	RO	RO	RO
x	1	1	-	n/a	RO	RO

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.TID3](#)==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [HSTR_EL2.T0](#)==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.TID3](#)==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [HSTR_EL2.T0](#)==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HCR.TID3](#)==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.
- If [HSTR.T0](#)==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

28/09/2017 08:16:24

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

MIDR, Main ID Register

The MIDR characteristics are:

Purpose

Provides identification information for the PE, including an implementer code for the device and a device ID number.

This register is part of the Identification registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register MIDR is architecturally mapped to AArch64 System register [MIDR_EL1](#).

AArch32 System register MIDR is architecturally mapped to External register [MIDR_EL1](#).

Some fields of the MIDR are IMPLEMENTATION DEFINED. For details of the values of these fields for a particular ARMv8 implementation, and any implementation-specific significance of these values, see the product documentation.

Attributes

MIDR is a 32-bit register.

Field descriptions

The MIDR bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Implementer								Variant				Architecture				PartNum								Revision							

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:

Hex representation	ASCII representation	Implementer
0x41	A	ARM Limited
0x42	B	Broadcom Corporation
0x43	C	Cavium Inc.
0x44	D	Digital Equipment Corporation
0x49	I	Infineon Technologies AG
0x4D	M	Motorola or Freescale Semiconductor Inc.
0x4E	N	NVIDIA Corporation
0x50	P	Applied Micro Circuits Corporation
0x51	Q	Qualcomm Inc.
0x56	V	Marvell International Ltd.
0x69	i	Intel Corporation

ARM can assign codes that are not published in this manual. All values not assigned by ARM are reserved and must not be used.

Variant, bits [23:20]

An IMPLEMENTATION DEFINED variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product.

Architecture, bits [19:16]

The permitted values of this field are:

Architecture	Meaning
0001	ARMv4
0010	ARMv4T
0011	ARMv5 (obsolete)
0100	ARMv5T
0101	ARMv5TE
0110	ARMv5TEJ
0111	ARMv6
1111	Architectural features are individually identified in the ID_* registers, see 'ID Identification registers, functional group' in the ARMv8 ARM, section K12.5.3G4.18.1.

All other values are reserved.

PartNum, bits [15:4]

An IMPLEMENTATION DEFINED primary part number for the device.

On processors implemented by ARM, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently.

Revision, bits [3:0]

An IMPLEMENTATION DEFINED revision number for the device.

Accessing the MIDR

This register can be read using MRC with the following syntax:

MRC <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c0, c0, 0	000	000	0000	1111	0000

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RO	n/a	RO
x	0	1	-	RO	RO	RO
x	1	1	-	n/a	RO	RO

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HSTR_EL2](#).T0==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2](#).T0==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HSTR](#).T0==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

28/0907/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMCCNTR, Performance Monitors Cycle Count Register

The PMCCNTR characteristics are:

Purpose

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. See 'Time as measured by the Performance Monitors cycle counter' in the ARMv8 ARM, section D5 for more information.

[PMCCFILTR](#) determines the modes and states in which the PMCCNTR can increment.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMCCNTR is architecturally mapped to AArch64 System register [PMCCNTR_EL0](#) when accessing as a 64-bit register.

AArch32 System register PMCCNTR is architecturally mapped to External register [PMCCNTR_EL0](#).

All counters are subject to any changes in clock frequency, including clock stopping caused by the WFI and WFE instructions. This means that it is CONSTRAINED UNPREDICTABLE whether or not PMCCNTR continues to increment when clocks are stopped by WFI and WFE instructions.

This register is in the Warm reset domain. On a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

Attributes

PMCCNTR is a 64-bit register that can also be accessed as a 32-bit value. If it is accessed as a 32-bit register, accesses read and write bits [31:0] and do not modify bits [63:32].

Field descriptions

The PMCCNTR bit assignments are:

When accessing as 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
																CCNT															

CCNT, bits [31:0]

Cycle count. Depending on the values of [PMCR](#).{LC,D}, this field increments in one of the following ways:

- Every processor clock cycle.
- Every 64th processor clock cycle.

Writing 1 to [PMCR](#).C sets this field to 0.

When accessing as a 64-bit register:

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

CCNT, bits [63:0]

Cycle count. Depending on the values of [PMCR](#).{LC,D}, this field increments in one of the following ways:

- Every processor clock cycle.
- Every 64th processor clock cycle.

Writing 1 to [PMCR](#).C sets this field to 0.

Accessing the PMCCNTR

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c13, 0	000	000	1001	1111	1101

This register can be read using MRRC with the following syntax:

```
MRRC <syntax>
```

This register can be written using MCRR with the following syntax:

```
MCRR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	coproc	CRm
p15, 0, <Rt>, <Rt2>, c9	0000	1111	1001

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
x	0	1	RW	RW	RW	RW
x	1	1	RW	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR.CR](#)==0, and [PMUSERENR.EN](#)==0, read accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_EL0.CR](#)==0, and [PMUSERENR_EL0.EN](#)==0, read accesses to this register from EL0 are trapped to EL1.
- If [PMUSERENR.EN](#)==0, write accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_EL0.EN](#)==0, write accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 :

- If [MDCR_EL2.TPM](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==1 && [HCR_EL2.TGE](#)==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure ~~write~~ accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and [SCR_EL3.NS](#)==1 :

- If [HDCR.TPM](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR.T9](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3.TPM](#)==1, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
[SysReg_v83A_xml-00bet4](#)

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMCEID0, Performance Monitors Common Event Identification register 0

The PMCEID0 characteristics are:

Purpose

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x000 to 0x01F

When the value of a bit in the register is 1 the corresponding common event is implemented or counted.

Note

ARM recommends that, if a **common** event is never counted, the value of the corresponding register bit is 0.

For more information about the common events and the use of the PMCEIDn registers see The section describing 'Event numbers and common events' in chapter D5 'The Performance Monitors Extension' of the ARM Architecture Reference Manual, for ARMv8-A architecture profile.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMCEID0 is architecturally mapped to AArch64 System register [PMCEID0_EL0\[31:0\]](#).

AArch32 System register PMCEID0 is architecturally mapped to External register [PMCEID0\[31:0\]](#).

Attributes

PMCEID0 is a 32-bit register.

Field descriptions

The PMCEID0 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																ID[31:0]															

ID[31:0], bits [31:0]

ID[n] corresponds to common event n.

For each bit:

ID[31:0]	Meaning
0	The common event is not implemented, or not counted.
1	The common event is implemented.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an **additional** common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing the PMCEID0

This register can be read using MRC with the following syntax:

MRC <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c12, 6	000	110	1001	1111	1100

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RO	RO	n/a	RO
x	0	1	RO	RO	RO	RO
x	1	1	RO	n/a	RO	RO

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR.EN](#)==0, read accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_EL0.EN](#)==0, read accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure read accesses to this register from EL0 **and EL1** are trapped to EL2.
- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2.TPM](#)==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure read accesses to this register from EL0 are trapped to EL2.
- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR.TPM](#)==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR.T9](#)==1, Non-secure read accesses to this register from EL0 **and EL1** are trapped to Hyp mode.

- If [HSTR.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3.TPM](#)==1, read accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMCEID1, Performance Monitors Common Event Identification register 1

The PMCEID1 characteristics are:

Purpose

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x020 to 0x03F.

When the value of a bit in the register is 1 the corresponding common event is implemented or counted.

Note

ARM recommends that, if a **common** event is never counted, the value of the corresponding register bit is 0.

For more information about the common events and the use of the PMCEIDn registers see The section describing 'Event numbers and common events' in chapter D5 'The Performance Monitors Extension' of the ARM Architecture Reference Manual, for ARMv8-A architecture profile.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMCEID1 is architecturally mapped to AArch64 System register [PMCEID1_EL0\[31:0\]](#).

AArch32 System register PMCEID1 is architecturally mapped to External register [PMCEID1\[31:0\]](#).

Attributes

PMCEID1 is a 32-bit register.

Field descriptions

The PMCEID1 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[63:32]																															

ID[63:32], bits [31:0]

ID[n] corresponds to common event n.

For each bit:

ID[63:32]	Meaning
0	The common event is not implemented, or not counted.
1	The common event is implemented.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an **additional** common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing the PMCEID1

This register can be read using MRC with the following syntax:

MRC <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c12, 7	000	111	1001	1111	1100

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RO	RO	n/a	RO
x	0	1	RO	RO	RO	RO
x	1	1	RO	n/a	RO	RO

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR.EN](#)==0, read accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_EL0.EN](#)==0, read accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure read accesses to this register from EL0 **and EL1** are trapped to EL2.
- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2.TPM](#)==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure read accesses to this register from EL0 are trapped to EL2.
- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR.TPM](#)==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR.T9](#)==1, Non-secure read accesses to this register from EL0 **and EL1** are trapped to Hyp mode.

- If [HSTR.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3.TPM](#)==1, read accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMCEID2, Performance Monitors Common Event Identification register 2

The PMCEID2 characteristics are:

Purpose

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x4000 to 0x401F.

When the value of a bit in the register is 1 the corresponding common event is implemented or counted.

Note

ARM recommends that, if a **common** event is never counted, the value of the corresponding register bit is 0.

For more information about the common events and the use of the PMCEIDn registers see The section describing 'Event numbers and common events' in chapter D5 'The Performance Monitors Extension' of the ARM Architecture Reference Manual, for ARMv8-A architecture profile.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMCEID2 is architecturally mapped to AArch64 System register [PMCEID0_EL0\[63:32\]](#).

AArch32 System register PMCEID2 is architecturally mapped to External register [PMCEID2\[63:32\]](#).

This register is introduced in ARMv8.1.

Attributes

PMCEID2 is a 32-bit register.

Field descriptions

The PMCEID2 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDhi[31:0]																															

IDhi[31:0], bits [31:0]

IDhi[n] corresponds to common event (0x4000 + n).

For each bit:

IDhi[31:0]	Meaning
0	The common event is not implemented, or not counted.
1	The common event is implemented.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an **additional** common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing the PMCEID2

This register can be read using MRC with the following syntax:

MRC <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c14, 4	000	100	1001	1111	1110

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RO	RO	n/a	RO
x	0	1	RO	RO	RO	RO
x	1	1	RO	n/a	RO	RO

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR.EN](#)==0, read accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_EL0.EN](#)==0, read accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure read accesses to this register from EL0 are trapped to EL2.
- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 :

- If [MDCR_EL2.TPM](#)==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==1 && [HCR_EL2.TGE](#)==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure read accesses to this register from EL0 are trapped to EL2.
- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and [SCR_EL3.NS](#)==1 :

- If [HDCR.TPM](#)==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR.T9](#)==1, Non-secure read accesses to this register from EL0 are trapped to Hyp mode.

- If `HSTR.T9==1`, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If `MDCR_EL3.TPM==1`, read accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg_v83A_xml-00bet4
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
SysReg_v83A_xml-00bet5

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMCEID3, Performance Monitors Common Event Identification register 3

The PMCEID3 characteristics are:

Purpose

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x4020 to 0x403F.

When the value of a bit in the register is 1 the corresponding common event is implemented or counted.

Note

ARM recommends that, if a **common** event is never counted, the value of the corresponding register bit is 0.

For more information about the common events and the use of the PMCEIDn registers see The section describing 'Event numbers and common events' in chapter D5 'The Performance Monitors Extension' of the ARM Architecture Reference Manual, for ARMv8-A architecture profile.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMCEID3 is architecturally mapped to AArch64 System register [PMCEID1_EL0\[63:32\]](#).

AArch32 System register PMCEID3 is architecturally mapped to External register [PMCEID3\[63:32\]](#).

This register is introduced in ARMv8.1.

Attributes

PMCEID3 is a 32-bit register.

Field descriptions

The PMCEID3 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDhi[63:32]																															

IDhi[63:32], bits [31:0]

IDhi[n] corresponds to common event (0x4000 + n).

For each bit:

IDhi[63:32]	Meaning
0	The common event is not implemented, or not counted.
1	The common event is implemented.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an **additional** common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing the PMCEID3

This register can be read using MRC with the following syntax:

MRC <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c14, 5	000	101	1001	1111	1110

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RO	RO	n/a	RO
x	0	1	RO	RO	RO	RO
x	1	1	RO	n/a	RO	RO

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR.EN](#)==0, read accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_EL0.EN](#)==0, read accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure read accesses to this register from EL0 are trapped to EL2.
- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 :

- If [MDCR_EL2.TPM](#)==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==1 && [HCR_EL2.TGE](#)==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure read accesses to this register from EL0 are trapped to EL2.
- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and [SCR_EL3.NS](#)==1 :

- If [HDCR.TPM](#)==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR.T9](#)==1, Non-secure read accesses to this register from EL0 are trapped to Hyp mode.

- If `HSTR.T9==1`, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If `MDCR_EL3.TPM==1`, read accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg_v83A_xml-00bet4
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
SysReg_v83A_xml-00bet5

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMCNTENCLR, Performance Monitors Count Enable Clear register

The PMCNTENCLR characteristics are:

Purpose

Disables the Cycle Count Register, [PMCCNTR](#), and any implemented event counters [PMEVCNTR<n>](#). Reading this register shows which counters are enabled.

PMCNTENCLR is used in conjunction with the [PMCNTENSET](#) register.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMCNTENCLR is architecturally mapped to AArch64 System register [PMCNTENCLR_EL0](#).

AArch32 System register PMCNTENCLR is architecturally mapped to External register [PMCNTENCLR_EL0](#).

This register is in the Warm reset domain. On a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

Attributes

PMCNTENCLR is a 32-bit register.

Field descriptions

The PMCNTENCLR bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P<n>, bit [n]																														

C, bit [31]

[PMCCNTR](#) disable bit. Disables the cycle counter register. Possible values are:

C	Meaning
0	When read, means the cycle counter is disabled. When written, has no effect.
1	When read, means the cycle counter is enabled. When written, disables the cycle counter.

P<n>, bit [n], for n = 0 to 30

Event counter disable bit for [PMEVCNTR<n>](#).

Bits [30:N] are RAZ/WI. When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN if EL2 is using AArch64 or in [HDCR](#).HPMN if EL2 is using AArch32. Otherwise, N is the value in [PMCR](#).N.

Possible values of each bit are:

P<n>	Meaning
0	When read, means that PMEVCNTR<n> is disabled. When written, has no effect.
1	When read, means that PMEVCNTR<n> is enabled. When written, disables PMEVCNTR<n> .

Accessing the PMCNTENCLR

This register can be read using MRC with the following syntax:

MRC <syntax>

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c12, 2	000	010	1001	1111	1100

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
x	0	1	RW	RW	RW	RW
x	1	1	RW	n/a	RW	RW

This table applies to all instructions that can access this register.

If EL2 is implemented, in Non-secure EL1 and EL0 modes, the value of [HDCR.HPMN](#) or [MDCR_EL2.HPMN](#) can change the behavior of accesses to PMCNTENCLR. See the description of the P<n> bit.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR.EN](#)==0, accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_EL0.EN](#)==0, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2.TPM](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure **write** accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR.HSTR.TPM](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR.T9](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TPM==1, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

<u>SysReg_v83A_xml-00bet4</u> <u>(old)</u>	htmldiff from-	<u>(new)</u>
	SysReg_v83A_xml-00bet4	<u>SysReg_v83A_xml-00bet5</u>

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMCNTENSET, Performance Monitors Count Enable Set register

The PMCNTENSET characteristics are:

Purpose

Enables the Cycle Count Register, [PMCCNTR](#), and any implemented event counters [PMEVCNTR<n>](#). Reading this register shows which counters are enabled.

PMCNTENSET is used in conjunction with the [PMCNTENCLR](#) register.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMCNTENSET is architecturally mapped to AArch64 System register [PMCNTENSET_EL0](#).

AArch32 System register PMCNTENSET is architecturally mapped to External register [PMCNTENSET_EL0](#).

This register is in the Warm reset domain. On a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

Attributes

PMCNTENSET is a 32-bit register.

Field descriptions

The PMCNTENSET bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P<n>, bit [n]																														

C, bit [31]

[PMCCNTR](#) enable bit. Enables the cycle counter register. Possible values are:

C	Meaning
0	When read, means the cycle counter is disabled. When written, has no effect.
1	When read, means the cycle counter is enabled. When written, enables the cycle counter.

P<n>, bit [n], for n = 0 to 30

Event counter enable bit for [PMEVCNTR<n>](#).

Bits [30:N] are RAZ/WI. When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN, if EL2 is using AArch64 or in [HDCR](#).HPMN if EL2 is using AArch32. Otherwise, N is the value in [PMCR](#).N.

Possible values of each bit are:

P<n>	Meaning
0	When read, means that PMEVCNTR<n> is disabled. When written, has no effect.
1	When read, means that PMEVCNTR<n> event counter is enabled. When written, enables PMEVCNTR<n> .

Accessing the PMCNTENSET

This register can be read using MRC with the following syntax:

MRC <syntax>

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c12, 1	000	001	1001	1111	1100

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
x	0	1	RW	RW	RW	RW
x	1	1	RW	n/a	RW	RW

This table applies to all instructions that can access this register.

If EL2 is implemented, in Non-secure EL1 and EL0 modes, the value of [HDCR](#).HPMN can change the behavior of accesses to PMCNTENSET. See the description of the P<n> bit.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR](#).EN==0, accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_EL0](#).EN==0, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HSTR_EL2](#).T9==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2](#).TPM==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2](#).T9==1, Non-secure **write** accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR](#).TPM==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR](#).T9==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TPM==1, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
[\(old\)](#)

htmldiff from-
SysReg_v83A_xml-00bet4

[\(new\)](#)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMCR is architecturally mapped to AArch64 System register [PMCR_EL0](#).

AArch32 System register PMCR bits [6:0] are architecturally mapped to External register [PMCR_EL0\[6:0\]](#).

This register is in the Warm reset domain. Some or all RW fields of this register have defined reset values. On a Warm or Cold reset these apply only if the PE resets into an Exception level that is using AArch32. Otherwise, on a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

Attributes

PMCR is a 32-bit register.

Field descriptions

The PMCR bit assignments are:

31	30	29	28	27	26	25	24	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				0	0	0	0	LC	DP	X	D	C	P	E	

IMP, bits [31:24]

Implementer code. This field is RO with an IMPLEMENTATION DEFINED value.

The implementer codes are allocated by ARM. Values have the same interpretation as bits [31:24] of the [MIDR](#).

IDCODE, bits [23:16]

Identification code. This field is RO with an IMPLEMENTATION DEFINED value.

Each implementer must maintain a list of identification codes that is specific to the implementer. A specific implementation is identified by the combination of the implementer code and the identification code.

N, bits [15:11]

Number of event counters. A RO field that indicates the number counters implemented. A value of 0b00000 in this field indicates that only the Cycle Count Register [PMCCNTR](#) is implemented.

The value of this field is the number of event counters implemented. This value is in the range of 0b00000, in which case only the [PMCCNTR](#) is implemented, to 0b11111, which indicates that the [PMCCNTR](#) and 31 event counters are implemented.

In an implementation that includes EL2, reads of this field from Non-secure EL1 and Non-secure EL0 return the value of [HDCR](#).HPMN if EL2 is using AArch32, or the value of [MDCR_EL2](#).HPMN if EL2 is using AArch64.

Bits [10:7]

Reserved, RES0.

LC, bit [6]

Long cycle counter enable. Determines which [PMCCNTR](#) bit generates an overflow recorded by [PMOVSRR](#)[31].

LC	Meaning
0	Cycle counter overflow on increment that changes PMCCNTR [31] from 1 to 0.
1	Cycle counter overflow on increment that changes PMCCNTR [63] from 1 to 0.

ARM deprecates use of PMCR.LC = 0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

DP, bit [5]

Disable cycle counter when event counting is prohibited. The possible values of this bit are:

DP	Meaning
0	PMCCNTR , if enabled, counts when event counting is prohibited.
1	PMCCNTR does not count when event counting is prohibited.

Counting events is never prohibited in Non-secure state. However, there are some restrictions on counting events in Secure state. For more information about the interaction between the Performance Monitors and EL3, see 'Interaction with EL3' in the ARMv8 ARM, section D5.5.1

When EL3 is not implemented, this field is RES0:

- When ARMv8.1-PMU is not implemented.
- When ARMv8.1-PMU is implemented, only if EL2 is not implemented.

Otherwise this field is RW.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

X, bit [4]

Enable export of events in an IMPLEMENTATION DEFINED event stream. The possible values of this bit are:

X	Meaning
0	Do not export events.
1	Export events where not prohibited.

This field enables the exporting of events over an event bus to another device, for example to an OPTIONAL trace macrocell. If the implementation does not include such an event bus then this field is RAZ/WI, otherwise it is an RW field.

In an implementation that includes an event bus, 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 0.

D, bit [3]

Clock divider. The possible values of this bit are:

D	Meaning
0	When enabled, PMCCNTR counts every clock cycle.
1	When enabled, PMCCNTR counts once every 64 clock cycles.

This bit is RW.

If PMCR.LC == 1, this bit is ignored and the cycle counter counts every clock cycle.

ARM deprecates use of PMCR.D = 1.

When this register has an architecturally-defined reset value, this field resets to 0.

C, bit [2]

Cycle counter reset. This bit is WO. The effects of writing to this bit are:

C	Meaning
0	No action.
1	Reset PMCCNTR to zero.

This bit is always RAZ.

Resetting [PMCCNTR](#) does not clear the [PMCCNTR](#) overflow bit to 0.

P, bit [1]

Event counter reset. This bit is WO. The effects of writing to this bit are:

P	Meaning
0	No action.
1	Reset all event counters accessible in the current EL, not including PMCCNTR , to zero.

This bit is always RAZ.

In Non-secure EL0 and EL1, if EL2 is implemented, a write of 1 to this bit does not reset event counters that [HDCR](#).HPMN or [MDCR_EL2](#).HPMN reserves for EL2 use.

In EL2 and EL3, a write of 1 to this bit resets all the event counters.

Resetting the event counters does not clear any overflow bits to 0.

E, bit [0]

Enable. The possible values of this bit are:

E	Meaning
0	All counters that are accessible at Non-secure EL1, including PMCCNTR , are disabled.
1	All counters that are accessible at Non-secure EL1 are enabled by PMCNTENSET .

This bit is RW.

If EL2 is implemented, this bit does not affect the operation of event counters that [HDCR](#).HPMN or [MDCR_EL2](#).HPMN reserves for EL2 use.

When this register has an architecturally-defined reset value, this field resets to 0.

Accessing the PMCR

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c12, 0	000	000	1001	1111	1100

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
x	0	1	RW	RW	RW	RW
x	1	1	RW	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR](#).EN==0, accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR](#).EL0.EN==0, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HSTR](#).EL2.T9==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR](#).EL2.TPM==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.
- If [MDCR](#).EL2.TPMCR==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HSTR](#).EL2.T9==1, Non-secure **write** accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR](#).TPM==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HDCR](#).TPMCR==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR](#).T9==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR](#).EL3.TPM==1, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMINTENCLR, Performance Monitors Interrupt Enable Clear register

The PMINTENCLR characteristics are:

Purpose

Disables the generation of interrupt requests on overflows from the Cycle Count Register, [PMCCNTR](#), and the event counters [PMEVCNTR<n>](#). Reading the register shows which overflow interrupt requests are enabled.

PMINTENCLR is used in conjunction with the [PMINTENSET](#) register.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMINTENCLR is architecturally mapped to AArch64 System register [PMINTENCLR_EL1](#).

AArch32 System register PMINTENCLR is architecturally mapped to External register [PMINTENCLR_EL1](#).

This register is in the Warm reset domain. On a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

Attributes

PMINTENCLR is a 32-bit register.

Field descriptions

The PMINTENCLR bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P<n>, bit [n]																														

C, bit [31]

[PMCCNTR](#) overflow interrupt request disable bit. Possible values are:

C	Meaning
0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
1	When read, means the cycle counter overflow interrupt request is enabled. When written, disables the cycle count overflow interrupt request.

P<n>, bit [n], for n = 0 to 30

Event counter overflow interrupt request disable bit for [PMEVCNTR<n>](#).

When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [HDCR](#).HPMN. Otherwise, N is the value in [PMCR](#).N.

Bits [30:N] are RAZ/WI.

Possible values are:

P<n>	Meaning
0	When read, means that the PMEVCNTR<n> event counter interrupt request is disabled. When written, has no effect.
1	When read, means that the PMEVCNTR<n> event counter interrupt request is enabled. When written, disables the PMEVCNTR<n> interrupt request.

Accessing the PMINTENCLR

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c14, 2	000	010	1001	1111	1110

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
x	0	1	-	RW	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

If EL2 is implemented, in Non-secure EL1 and EL0 modes, the value of [HDCR](#).HPMN can change the behavior of accesses to PMINTENCLR. See the description of the P<n> bit.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HSTR_EL2](#).T9==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2](#).TPM==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2](#).T9==1, Non-secure **write** accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR](#).TPM==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.
- If [HSTR](#).T9==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TPM==1, accesses to this register from EL1 and EL2 are trapped to EL3.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
[\(old\)](#)

htmldiff from-
SysReg_v83A_xml-00bet4

[\(new\)](#)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMINTENSET, Performance Monitors Interrupt Enable Set register

The PMINTENSET characteristics are:

Purpose

Enables the generation of interrupt requests on overflows from the Cycle Count Register, [PMCCNTR](#), and the event counters [PMEVCNTR<n>](#). Reading the register shows which overflow interrupt requests are enabled.

PMINTENSET is used in conjunction with the [PMINTENCLR](#) register.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMINTENSET is architecturally mapped to AArch64 System register [PMINTENSET_EL1](#).

AArch32 System register PMINTENSET is architecturally mapped to External register [PMINTENSET_EL1](#).

This register is in the Warm reset domain. On a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

Attributes

PMINTENSET is a 32-bit register.

Field descriptions

The PMINTENSET bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P<n>, bit [n]																														

C, bit [31]

[PMCCNTR](#) overflow interrupt request enable bit. Possible values are:

C	Meaning
0	When read, means the cycle counter overflow interrupt request is disabled. When written, has no effect.
1	When read, means the cycle counter overflow interrupt request is enabled. When written, enables the cycle count overflow interrupt request.

P<n>, bit [n], for n = 0 to 30

Event counter overflow interrupt request enable bit for [PMEVCNTR<n>](#).

When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [HDCR](#).HPMN. Otherwise, N is the value in [PMCR](#).N.

Bits [30:N] are RAZ/WI.

Possible values are:

P<n>	Meaning
0	When read, means that the PMEVCNTR<n> event counter interrupt request is disabled. When written, has no effect.
1	When read, means that the PMEVCNTR<n> event counter interrupt request is enabled. When written, enables the PMEVCNTR<n> interrupt request.

Accessing the PMINTENSET

This register can be read using MRC with the following syntax:

MRC <syntax>

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c14, 1	000	001	1001	1111	1110

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
x	0	1	-	RW	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

If EL2 is implemented, in Non-secure EL1 and EL0 modes, the value of [HDCR](#).HPMN can change the behavior of accesses to PMINTENSET. See the description of the P<n> bit.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HSTR_EL2](#).T9==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2](#).TPM==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2](#).T9==1, Non-secure **write** accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR](#).TPM==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.
- If [HSTR](#).T9==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TPM==1, accesses to this register from EL1 and EL2 are trapped to EL3.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
[\(old\)](#)

htmldiff from-
SysReg_v83A_xml-00bet4

[\(new\)](#)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMOVSr, Performance Monitors Overflow Flag Status Register

The PMOVSr characteristics are:

Purpose

Contains the state of the overflow bit for the Cycle Count Register, [PMCCNTR](#), and each of the implemented event counters [PMEVCNTR<n>](#). Writing to this register clears these bits.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMOVSr is architecturally mapped to AArch64 System register [PMOVSCLR_EL0](#).

AArch32 System register PMOVSr is architecturally mapped to External register [PMOVSCLR_EL0](#).

This register is in the Warm reset domain. On a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

Attributes

PMOVSr is a 32-bit register.

Field descriptions

The PMOVSr bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P<n>, bit [n]																														

C, bit [31]

[PMCCNTR](#) overflow bit. Possible values are:

C	Meaning
0	When read, means the cycle counter has not overflowed. When written, has no effect.
1	When read, means the cycle counter has overflowed. When written, clears the overflow bit to 0.

[PMCR](#).LC controls whether an overflow is detected from [PMCCNTR](#)[31] or from [PMCCNTR](#)[63].

P<n>, bit [n], for n = 0 to 30

Event counter overflow clear bit for [PMEVCNTR<n>](#).

Bits [30:N] are RAZ/WI. When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN if EL2 is using AArch64 or in [HDCR](#).HPMN if EL2 is using AArch32. Otherwise, N is the value in [PMCR](#).N.

Possible values of each bit are:

P<n>	Meaning
0	When read, means that PMEVCNTR<n> has not overflowed. When written, has no effect.
1	When read, means that PMEVCNTR<n> has overflowed. When written, clears the PMEVCNTR<n> overflow bit to 0.

Accessing the PMOVSr

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c12, 3	000	011	1001	1111	1100

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
x	0	1	RW	RW	RW	RW
x	1	1	RW	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR.EN](#)==0, accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_EL0.EN](#)==0, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 :

- If [MDCR_EL2.TPM](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==1 && [HCR_EL2.TGE](#)==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure **write** accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and [SCR_EL3.NS](#)==1 :

- If [HDCR.TPM](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR.T9](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TPM==1, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg_v83A_xml-00bet4	htmldiff from-	(new)
(old)	SysReg_v83A_xml-00bet4	SysReg_v83A_xml-00bet5

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMOVSSET, Performance Monitors Overflow Flag Status Set register

The PMOVSSET characteristics are:

Purpose

Sets the state of the overflow bit for the Cycle Count Register, [PMCCNTR](#), and each of the implemented event counters [PMEVCNTR<n>](#).

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMOVSSET is architecturally mapped to AArch64 System register [PMOVSSET_EL0](#).

AArch32 System register PMOVSSET is architecturally mapped to External register [PMOVSSET_EL0](#).

This register is in the Warm reset domain. On a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

Attributes

PMOVSSET is a 32-bit register.

Field descriptions

The PMOVSSET bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C	P<n>, bit [n]																														

C, bit [31]

[PMCCNTR](#) overflow bit. Possible values are:

C	Meaning
0	When read, means the cycle counter has not overflowed. When written, has no effect.
1	When read, means the cycle counter has overflowed. When written, sets the overflow bit to 1.

P<n>, bit [n], for n = 0 to 30

Event counter overflow set bit for [PMEVCNTR<n>](#).

Bits [30:N] are RAZ/WI. When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN if EL2 is using AArch64 or in [HDCR](#).HPMN if EL2 is using AArch32. Otherwise, N is the value in [PMCR](#).N.

Possible values are:

P<n>	Meaning
0	When read, means that PMEVCNTR<n> has not overflowed. When written, has no effect.
1	When read, means that PMEVCNTR<n> has overflowed. When written, sets the PMEVCNTR<n> overflow bit to 1.

Accessing the PMOVSSET

This register can be read using MRC with the following syntax:

MRC <syntax>

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c14, 3	000	011	1001	1111	1110

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
x	0	1	RW	RW	RW	RW
x	1	1	RW	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR.EN](#)=0, accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_EL0.EN](#)=0, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)=1 && [HCR_EL2.E2H](#)=0 :

- If [HSTR_EL2.T9](#)=1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)=1 :

- If [MDCR_EL2.TPM](#)=1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)=1 && [HCR_EL2.E2H](#)=1 && [HCR_EL2.TGE](#)=0 :

- If [HSTR_EL2.T9](#)=1, Non-secure **write** accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and [SCR_EL3.NS](#)=1 :

- If [HDCR.TPM](#)=1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR.T9](#)=1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3.TPM](#)=1, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

[SysReg_v83A_xml-00bet4](#)
[\(old\)](#)

htmldiff from-
SysReg_v83A_xml-00bet4

[\(new\)](#)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMSELR, Performance Monitors Event Counter Selection Register

The PMSELR characteristics are:

Purpose

Selects the current event counter [PMEVCNTR<n>](#) or the cycle counter, CCNT.

PMSELR is used in conjunction with [PMXEVTYPER](#) to determine the event that increments a selected event counter, and the modes and states in which the selected counter increments.

It is also used in conjunction with [PMXEVNTR](#), to determine the value of a selected event counter.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMSELR is architecturally mapped to AArch64 System register [PMSELR_EL0](#).

This register is in the Warm reset domain. On a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

Attributes

PMSELR is a 32-bit register.

Field descriptions

The PMSELR bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SEL				

Bits [31:5]

Reserved, RES0.

SEL, bits [4:0]

Selects event counter, [PMEVCNTR<n>](#), where n is the value held in this field. This value identifies which event counter is accessed when a subsequent access to [PMXEVTYPER](#) or [PMXEVNTR](#) occurs.

This field can take any value from 0 (0b00000) to (PMCR.N)-1, or 31 (0b11111).

When PMSELR.SEL is 0b11111, it selects the cycle counter and:

- A read of the [PMXEVTYPER](#) returns the value of [PMCCFILTR](#).
- A write of the [PMXEVTYPER](#) writes to [PMCCFILTR](#).
- A read or write of [PMXEVNTR](#) has CONSTRAINED UNPREDICTABLE effects, that can be one of the following:
 - Access to [PMXEVNTR](#) is UNDEFINED.
 - Access to [PMXEVNTR](#) behaves as a NOP.
 - Access to [PMXEVNTR](#) behaves as if the register is RAZ/WI.
 - Access to [PMXEVNTR](#) behaves as if the PMSELR.SEL field contains an UNKNOWN value.

If this field is set to a value greater than or equal to the number of implemented counters, but not equal to 31:

If this field is set to a value greater than or equal to the number of implemented counters, but not equal to 31, the results of access to `PMXEVTYPER` or `PMXEVNTR` are CONSTRAINED UNPREDICTABLE, and can be one of the following:

- Access to `PMXEVTYPER` or `PMXEVNTR` behaves as if the register is RAZ/WI.
- Access to `PMXEVTYPER` or `PMXEVNTR` behaves as if the `PMSELR.SEL` field contains an UNKNOWN value.
- Access to `PMXEVTYPER` or `PMXEVNTR` behaves as if the `PMSELR.SEL` field contains `0b11111`.
- Direct reads of this field return an UNKNOWN value.
- Access to `PMXEVTYPER` or `PMXEVNTR` is UNDEFINED.
- The results of access to `PMXEVTYPER` or `PMXEVNTR` are behaves as a NOP. CONSTRAINED UNPREDICTABLE, and can be one of the following:
 - Access to `PMXEVTYPER` or `PMXEVNTR` is UNDEFINED.
 - Access to `PMXEVTYPER` or `PMXEVNTR` behaves as a NOP.
 - Access to `PMXEVTYPER` or `PMXEVNTR` behaves as if the register is RAZ/WI.
 - Access to `PMXEVTYPER` or `PMXEVNTR` behaves as if the `PMSELR.SEL` field contains an UNKNOWN value.
 - Access to `PMXEVTYPER` or `PMXEVNTR` behaves as if the `PMSELR.SEL` field contains `0b11111`.

Direct reads of this field return an UNKNOWN value.

Accessing the PMSELR

This register can be read using MRC with the following syntax:

MRC <syntax>

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	ope1	ope2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c12, 5	000	101	1001	1111	1100

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
x	0	1	RW	RW	RW	RW
x	1	1	RW	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If `PMUSERENR.EN`==0, and `PMUSERENR.ER`==0, accesses to this register from EL0 are trapped to Undefined mode.
- If `PMUSERENR_EL0.EN`==0, and `PMUSERENR_EL0.ER`==0, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and `SCR_EL3.NS`==1 && `HCR_EL2.E2H`==0:

- If `HSTR_EL2.T9`==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and `SCR_EL3.NS`==1:

- If `MDCR_EL2.TPM==1`, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and `SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0`:

- If `HSTR_EL2.T9==1`, Non-secure write accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and `SCR_EL3.NS==1`:

- If `HDCR.TPM==1`, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If `HSTR.T9==1`, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64:

- If `MDCR_EL3.TPM==1`, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

Accessing the PMSELR

This register can be read using MRC with the following syntax:

MRC <syntax>

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c12, 5	000	101	1001	1111	1100

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
x	0	1	RW	RW	RW	RW
x	1	1	RW	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If `PMUSERENR.EN==0`, and `PMUSERENR.ER==0`, accesses to this register from EL0 are trapped to Undefined mode.
- If `PMUSERENR_EL0.EN==0`, and `PMUSERENR_EL0.ER==0`, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and `SCR_EL3.NS==1 && HCR_EL2.E2H==0`:

- If `HSTR_EL2.T9==1`, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and `SCR_EL3.NS==1`:

- If `MDCR_EL2.TPM==1`, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and `SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0`:

- If [HSTR_EL2.T9==1](#), Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and [SCR_EL3.NS==1](#) :

- If [HDCR.TPM==1](#), Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR.T9==1](#), Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3.TPM==1](#), accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMSWINC, Performance Monitors Software Increment register

The PMSWINC characteristics are:

Purpose

Increments a counter that is configured to count the Software increment event, event 0x00. For more information, see 'SW_INCR' in the ARMv8 ARM, section D5.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMSWINC is architecturally mapped to AArch64 System register [PMSWINC_EL0](#).

AArch32 System register PMSWINC is architecturally mapped to External register [PMSWINC_EL0](#).

Attributes

PMSWINC is a 32-bit register.

Field descriptions

The PMSWINC bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	P<n>, bit [n]																														

Bit [31]

Reserved, RES0.

P<n>, bit [n], for n = 0 to 30

Event counter software increment bit for [PMEVCNTR<n>](#).

Bits [30:N] are WI.

When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [MDCR_EL2](#).HPMN if EL2 is using AArch64 or in [HDCR](#).HPMN if EL2 is using AArch32. Otherwise, N is the value in [PMCR](#).N.

The effects of writing to this bit are:

P<n>	Meaning
0	No action. The write to this bit is ignored.
1	If PMEVCNTR<n> is enabled and configured to count the software increment event, increments PMEVCNTR<n> by 1. If PMEVCNTR<n> is disabled, or not configured to count the software increment event, the write to this bit is ignored.

Accessing the PMSWINC

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c12, 4	000	100	1001	1111	1100

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	WO	WO	n/a	WO
x	0	1	WO	WO	WO	WO
x	1	1	WO	n/a	WO	WO

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR.EN](#)==0, and [PMUSERENR.SW](#)==0, write accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_EL0.EN](#)==0, and [PMUSERENR_EL0.SW](#)==0, write accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure write accesses to this register from EL0 ~~and EL1~~ are trapped to EL2.
- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 :

- If [MDCR_EL2.TPM](#)==1, Non-secure write accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==1 && [HCR_EL2.TGE](#)==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure write accesses to this register from EL0 ~~and EL1~~ are trapped to EL2.
- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and [SCR_EL3.NS](#)==1 :

- If [HDCR.TPM](#)==1, Non-secure write accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR.T9](#)==1, Non-secure write accesses to this register from EL0 ~~and EL1~~ are trapped to Hyp mode.
- If [HSTR.T9](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3.TPM](#)==1, write accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

<u>SysReg_v83A_xml-00bet4</u> <u>(old)</u>	htmldiff from-	<u>(new)</u> <u>SysReg_v83A_xml-00bet5</u>
---	----------------	---

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMUSERENR, Performance Monitors User Enable Register

The PMUSERENR characteristics are:

Purpose

Enables or disables User mode access to the Performance Monitors.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMUSERENR is architecturally mapped to AArch64 System register [PMUSERENR_ELO](#).

This register is in the Warm reset domain. Some or all RW fields of this register have defined reset values. On a Warm or Cold reset these apply only if the PE resets into an Exception level that is using AArch32. Otherwise, on a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

Attributes

PMUSERENR is a 32-bit register.

Field descriptions

The PMUSERENR bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ER	CR	SW	EN

Bits [31:4]

Reserved, RES0.

ER, bit [3]

Event counter read trap control:

ER	Meaning
0	EL0 reads of the PMXVCNTR and PMEVCNTR<n> , and EL0 read/write access to the PMSELR , are trapped to Undefined mode if PMUSERENR.EN is also 0.
1	This control does not cause any instructions to be trapped.

When this register has an architecturally-defined reset value, this field resets to 0.

CR, bit [2]

Cycle counter read trap control:

CR	Meaning
0	EL0 reads of the PMCCNTR are trapped to Undefined mode if PMUSERENR.EN is also 0.
1	This control does not cause any instructions to be trapped.

When this register has an architecturally-defined reset value, this field resets to 0.

SW, bit [1]

Software increment write trap control:

SW	Meaning
0	EL0 writes to the PMSWINC are trapped to Undefined mode if PMUSERENR.EN is also 0.
1	This control does not cause any instructions to be trapped.

When this register has an architecturally-defined reset value, this field resets to 0.

EN, bit [0]

Traps EL0 accesses to the Performance Monitors registers to Undefined mode:

EN	Meaning
0	EL0 accesses to the Performance Monitors registers are trapped to Undefined mode, unless enabled by one of PMUSERENR.{ER, CR, SW}.
1	This control does not cause any instructions to be trapped. Software can access all PMU registers at EL0.

Note

- The PMUSERENR register is always RO at EL0 and not trapped by this bit.
- EL0 cannot read or write [PMINTENSET](#) and [PMINTENCLR](#).

When this register has an architecturally-defined reset value, this field resets to 0.

Accessing the PMUSERENR

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c14, 0	000	000	1001	1111	1110

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RO	RW	n/a	RW
x	0	1	RO	RW	RW	RW
x	1	1	RO	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HSTR_EL2](#).T9==1, Non-secure **read** accesses to this register from EL0 ~~and EL1~~ are trapped to EL2.
- If [HSTR_EL2](#).T9==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2](#).TPM==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2](#).T9==1, Non-secure **readwrite** accesses to this register from ~~EL0~~**EL1** are trapped to EL2.
- If [HSTR_EL2](#).T9==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR](#).TPM==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR](#).T9==1, Non-secure **read** accesses to this register from EL0 ~~and EL1~~ are trapped to Hyp mode.
- If [HSTR](#).T9==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TPM==1, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMXEVCNTR, Performance Monitors Selected Event Count Register

The PMXEVCNTR characteristics are:

Purpose

Reads or writes the value of the selected event counter, [PMEVCNTR<n>](#). [PMSELR](#).SEL determines which event counter is selected.

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMXEVCNTR is architecturally mapped to AArch64 System register [PMXEVCNTR_EL0](#).

This register is in the Warm reset domain. On a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

Attributes

PMXEVCNTR is a 32-bit register.

Field descriptions

The PMXEVCNTR bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PMEVCNTR<n>																															

PMEVCNTR<n>, bits [31:0]

Value of the selected event counter, [PMEVCNTR<n>](#), where n is the value stored in [PMSELR](#).SEL.

Accessing the PMXEVCNTR

This register can be read using MRC with the following syntax:

MRC <syntax>

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c13, 2	000	010	1001	1111	1101

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
x	0	1	RW	RW	RW	RW
x	1	1	RW	n/a	RW	RW

This table applies to all instructions that can access this register.

If [PMSELR](#).SEL is greater than or equal to the number of accessible counters then reads and writes of PMXEVNTR are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if [PMSELR](#).SEL has an UNKNOWN value less than the number of counters accessible at the current Exception level and Security state.
- Accesses to the register behave as if [PMSELR](#).SEL is 31.
- In Non-secure state, for an access from PL1 or a permitted access from PL0, if [PMSELR](#).SEL, or [PMSELR_EL0](#).SEL if EL1 is using AArch64, is greater than or equal to the number of accessible counters but is less than the number of implemented counters, the register access is trapped to EL2. Accesses from PL0 are permitted when:
 - EL1 is using AArch32 and the value of [PMUSERENR](#).EN is 1.
 - EL1 is using AArch64 and the value of [PMUSERENR_EL0](#).EN is 1.

Note

In an implementation that includes EL2, in Non-secure state at EL0 and EL1:

- If EL2 is using AArch32, [HDCR](#).HPMN identifies the number of accessible counters.
- If EL2 is using AArch64, [MDCR_EL2](#).HPMN identifies the number of accessible counters.

Otherwise, the number of accessible counters is the number of implemented counters.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR](#).EN==0, and [PMUSERENR](#).ER==0, read accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR](#).EN==0, write accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_EL0](#).EN==0, and [PMUSERENR_EL0](#).ER==0, read accesses to this register from EL0 are trapped to EL1.
- If [PMUSERENR_EL0](#).EN==0, write accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HSTR_EL2](#).T9==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2](#).TPM==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2](#).T9==1, Non-secure **write** accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR](#).TPM==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR](#).T9==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TPM==1, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMXEVTYPER, Performance Monitors Selected Event Type Register

The PMXEVTYPER characteristics are:

Purpose

When [PMSELR.SEL](#) selects an event counter, this accesses a [PMEVTYPER<n>](#) register. When [PMSELR.SEL](#) selects the cycle counter, this accesses [PMCCFILTR](#).

This register is part of the Performance Monitors registers functional group.

Configuration

There is one instance of this register that is used in both Secure and Non-secure states.

AArch32 System register PMXEVTYPER is architecturally mapped to AArch64 System register [PMXEVTYPER_ELO](#).

When the value of [PMSELR.SEL](#) is 31, to select the cycle counter, RW fields in this register have defined reset values that apply only when the PE resets into an Exception level that is using AArch32. See [PMCCFILTR](#) for the reset values.

Otherwise, RW fields in this register reset to IMPLEMENTATION DEFINED values that might be UNKNOWN. This applies whenever [PMSELR.SEL](#) selects an event counter.

Attributes

PMXEVTYPER is a 32-bit register.

Field descriptions

The PMXEVTYPER bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Event type register or PMCCFILTR																															

Bits [31:0]

Event type register or [PMCCFILTR](#).

When [PMSELR.SEL](#) == 31, this register accesses [PMCCFILTR](#).

Otherwise, this register accesses [PMEVTYPER<n>](#) where n is the value in [PMSELR.SEL](#).

Accessing the PMXEVTYPER

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c9, c13, 1	000	001	1001	1111	1101

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
x	0	1	RW	RW	RW	RW
x	1	1	RW	n/a	RW	RW

This table applies to all instructions that can access this register.

If [PMSELR.SEL](#) is greater than or equal to the number of accessible counters then reads and writes of PMXEVTYPYPER are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if [PMSELR.SEL](#) has an UNKNOWN value less than the number of counters accessible at the current Exception level and Security state.
- Accesses to the register behave as if [PMSELR.SEL](#) is 31.
- In Non-secure state, for an access from PL1 or a permitted access from PL0, if [PMSELR.SEL](#), or [PMSELR_ELO.SEL](#) if EL1 is using AArch64, is greater than or equal to the number of accessible counters but is less than the number of implemented counters, the register access is trapped to EL2. Accesses from PL0 are permitted when:
 - EL1 is using AArch32 and the value of [PMUSERENR.EN](#) is 1.
 - EL1 is using AArch64 and the value of [PMUSERENR_ELO.EN](#) is 1.

Note

In an implementation that includes EL2, in Non-secure state at EL0 and EL1:

- If EL2 is using AArch32, [HDCR.HPMN](#) identifies the number of accessible counters.
- If EL2 is using AArch64, [MDCR_EL2.HPMN](#) identifies the number of accessible counters.

Otherwise, the number of accessible counters is the number of implemented counters.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR.EN](#)==0, accesses to this register from EL0 are trapped to Undefined mode.
- If [PMUSERENR_ELO.EN](#)==0, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 :

- If [MDCR_EL2.TPM](#)==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==1 && [HCR_EL2.TGE](#)==0 :

- If [HSTR_EL2.T9](#)==1, Non-secure **write** accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HDCR](#).TPM==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HSTR](#).T9==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TPM==1, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/0907/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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 CPSR.F or CPSR.A bits can be modified when SCR.NS==1.

This register is part of the Security registers functional group.

Configuration

This register is only accessible in Secure state.

AArch32 System register SCR can be mapped to AArch64 System register [SCR_EL3](#), but this is not architecturally mandated.

Some or all RW fields of this register have defined reset values. These apply whenever the register is accessible. This means they apply when the PE resets into EL3 using AArch32.

Attributes

SCR is a 32-bit register.

Field descriptions

The SCR bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TERR	0	TW	ETW	0	0	SIF	HCE	SCD	nET	AW	FW	EA	FIQ	IRQ	NS

Bits [31:16]

Reserved, RES0.

TERR, bit [15]

Trap Error record accesses. If the RAS Extension is implemented, the possible values of this bit are:

TERR	Meaning
0	Accesses to the ER* registers from modes EL1 and EL2 do not generate a Monitor Trap exception. EL3.
1	Accesses to the ER* registers from modes EL1 other than EL2 Monitor mode generate a Monitor Trap exception. exception to EL3.

This bit resets to 0 on Warm reset.

When the RAS Extension is not implemented, this field is RES0.

Bit [14]

Reserved, RES0.

TWE, bit [13]

Traps WFE instructions to Monitor mode.

TWE	Meaning
0	This control does not cause any instructions to be trapped.
1	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 of 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 this register has an architecturally-defined reset value, this field resets to 0.

TWI, bit [12]

Traps WFI instructions to Monitor mode.

TWI	Meaning
0	This control does not cause any instructions to be trapped.
1	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 of 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 this register has an architecturally-defined reset value, 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
0	Secure state instruction fetches from Non-secure memory are permitted.
1	Secure state instruction fetches from Non-secure memory are not permitted.

This bit is permitted to be cached in a TLB.

When this register has an architecturally-defined reset value, this field resets to 0.

HCE, bit [8]

Hypervisor Call instruction enable. Enables EL2 and Non-secure EL1 execution of HVC instructions.

HCE	Meaning
0	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.
1	HVC instructions are enabled at EL2 and Non-secure EL1.

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.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

SCD, bit [7]

Secure Monitor Call disable. Disables SMC instructions.

SCD	Meaning
0	SMC instructions are enabled.
1	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.

When this register has an architecturally-defined reset value, this field resets to 0.

nET, bit [6]

Not Early Termination. This bit disables early termination. The possible values of this bit are:

nET	Meaning
0	Early termination permitted. Execution time of data operations can depend on the data values.
1	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.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it 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 [CPSR.A](#) masks an External abort taken from Non-secure state, and the possible values of this bit are:

AW	Meaning
0	External aborts taken from Non-secure state are not masked by CPSR.A , and are taken to EL3.
1	External aborts taken from Secure state are masked by CPSR.A . External aborts taken from either Security state are masked by CPSR.A . When CPSR.A is 0, the abort is taken to EL3.

When SCR.EA is 0 or [HCR.AMO](#) is 1, this bit has no effect.

When this register has an architecturally-defined reset value, 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 [CPSR.F](#) masks an FIQ interrupt taken from Non-secure state, and the possible values of this bit are:

FW	Meaning
0	An FIQ taken from Non-secure state is not masked by CPSR.F , and is taken to EL3.
1	An FIQ taken from Secure state is masked by CPSR.F . An FIQ taken from either Security state is masked by CPSR.F . When CPSR.F is 0, the FIQ is taken to EL3.

When SCR.FIQ is 0 or [HCR.FMO](#) is 1, this bit has no effect.

When this register has an architecturally-defined reset value, this field resets to 0.

EA, bit [3]

External Abort handler. This bit controls which mode takes External aborts. The possible values of this bit are:

EA	Meaning
0	External aborts taken to Abort mode.
1	External aborts taken to Monitor mode.

When this register has an architecturally-defined reset value, this field resets to 0.

FIQ, bit [2]

FIQ handler. This bit controls which mode takes FIQ exceptions. The possible values of this bit are:

FIQ	Meaning
0	FIQs taken to FIQ mode.
1	FIQs taken to Monitor mode.

When this register has an architecturally-defined reset value, this field resets to 0.

IRQ, bit [1]

IRQ handler. This bit controls which mode takes IRQ exceptions. The possible values of this bit are:

IRQ	Meaning
0	IRQs taken to IRQ mode.
1	IRQs taken to Monitor mode.

When this register has an architecturally-defined reset value, 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
0	PE is in Secure state.
1	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.

When this register has an architecturally-defined reset value, this field resets to 0.

Accessing the SCR

This register can be read using MRC with the following syntax:

MRC <syntax>

This register can be written using MCR with the following syntax:

MCR <syntax>

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c1, c1, 0	000	000	0001	1111	0001

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	RW
x	0	1	-	-	-	RW
x	1	1	-	n/a	-	RW

This table applies to all instructions that can access this register.

If EL3 is implemented and is using AArch64, any read or write to SCR from Secure EL1 using AArch32 is trapped as an exception to EL3.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HSTR_EL2.T1](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2.T1](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HSTR.T1](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

28/09/2017 08:16:24

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

SCTLR, System Control Register

The SCTLR characteristics are:

Purpose

Provides the top level control of the system, including its memory system.

This register is part of the Other system control registers functional group.

Configuration

AArch32 System register SCTLR is architecturally mapped to AArch64 System register [SCTLR_EL1](#).

When EL3 is using AArch32, write access to SCTLR(S) is disabled when the CP15SDISABLE signal is asserted HIGH.

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.

Some or all RW fields of this register have defined reset values. These apply only if the PE resets into an Exception level that is using AArch32. If the PE resets into EL3 using AArch32 they apply only to the Secure instance of the register. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

Attributes

SCTLR is a 32-bit register.

Field descriptions

The SCTLR bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																									
0	TE	A	F	E	T	R	E	0	0	E	E	0	S	P	A	N	1	0	U	W	X	N	W	X	N	h	T	W	E	0	h	T	W	I	0	0	V	I	1	0	0	S	E	D	I	T	D	U	N	K	C	P	1	5	B	E	N	L	S	M	A	O	E	h	T	L	S	M	D	C	A	M

Bit [31]

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
0	Exceptions, including reset, taken to A32 state.
1	Exceptions, including reset, taken to T32 state.

When this register has an architecturally-defined reset value, this field resets to an IMPLEMENTATION DEFINED choice between:

- 0.
- A value determined by an input configuration signal.

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
0	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.
1	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.

When this register has an architecturally-defined reset value, 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
0	TEX remap disabled. TEX[2:0] are used, with the C and B bits, to describe the memory region attributes.
1	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.

When this register has an architecturally-defined reset value, 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
0	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.
1	Big-endian. PSTATE.E is set/cleared to 1/0 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.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to an IMPLEMENTATION DEFINED choice between:

- 0.
- A value determined by an input configuration signal.

Bit [24]

Reserved, RES0.

SPAN, bit [23]

In ARMv8.3, ARMv8.2 and ARMv8.1:

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
0	CPSR .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.
1	The value of CPSR .PAN is left unchanged on taking an exception.

In ARMv8.0:

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
0	This control has no effect on memory access permissions.
1	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.

When this register has an architecturally-defined reset value, 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
0	This control has no effect on memory access permissions.
1	Any region that is writable in the PL1&0 translation regime is forced to XN for accesses from software executing at PL1 or PL0.

The WXN bit is permitted to be cached in a TLB.

When this register has an architecturally-defined reset value, this field resets to 0.

nTWE, bit [18]

Traps EL0 execution of WFE instructions to Undefined mode.

nTWE	Meaning
0	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.
1	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 of 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 this register has an architecturally-defined reset value, this field resets to 1.

Bit [17]

Reserved, RES0.

nTWI, bit [16]

Traps EL0 execution of WFI instructions to Undefined mode.

nTWI	Meaning
0	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.
1	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 of 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 this register has an architecturally-defined reset value, 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
0	Normal exception vectors. Base address is held in VBAR .
1	High exception vectors (Hivecs), base address 0xFFFF0000. This base address cannot be remapped.

When this register has an architecturally-defined reset value, this field resets to an IMPLEMENTATION DEFINED choice between:

- 0.
- A value determined by an input configuration signal.

I, bit [12]

Instruction access Cacheability control, for accesses at EL1 and EL0:

I	Meaning
0	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.
1	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 Non-secure EL1 and Non-secure 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.

When this register has an architecturally-defined reset value, this field resets to 0.

Bit [11]

Reserved, RES1.

Bits [10:9]

Reserved, RES0.

SED, bit [8]

SETEND instruction disable. Disables SETEND instructions at PL0 and PL1.

SED	Meaning
0	SETEND instruction execution is enabled at PL0 and PL1.
1	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.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

ITD, bit [7]

IT Disable. Disables some uses of IT instructions at PL1 and PL0.

ITD	Meaning
0	All IT instruction functionality is enabled at PL1 and PL0.
1	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: <div> <div>11xxxxxxxxxxxx</div> <div>All 32-bit instructions, and the 16-bit instructions B, UDF, SVC, LDM, and STM.</div> </div> <div> <div>1011xxxxxxxxxxxx</div> <div>All instructions in 'Miscellaneous 16-bit instructions' in the ARMv8 ARM, section F3.2.5.</div> </div> <div> <div>10100xxxxxxxxxxx</div> <div>ADD Rd, PC, #imm</div> </div> <div> <div>01001xxxxxxxxxxx</div> <div>LDR Rd, [PC, #imm]</div> </div> <div> <div>0100x1xxx1111xxx</div> <div>ADD Rdn, PC; CMP Rn, PC; MOV Rd, PC; BX PC; BLX PC.</div> </div> <div> <div>010001xx1xxxx111</div> <div>ADD PC, Rm; CMP PC, Rm; MOV PC, Rm. This pattern also covers UNPREDICTABLE cases with BLX Rn.</div> </div> <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. 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' in the ARMv8 ARM, section E1.2.4.

ITD is optional, but if it is implemented in the SCTLR then it must also be implemented in the [SCTLR_EL1](#). If it is not implemented then this bit is RAZ/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

UNK, bit [6]

Writes to this bit are IGNORED. Reads of this bit return an UNKNOWN value.

CP15BEN, bit [5]

System instruction memory barrier enable. Enables accesses to the DMB, DSB, and ISB System instructions in the (coproc==1111) encoding space from PL1 and PL0:

CP15BEN	Meaning
0	PL0 and PL1 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is UNDEFINED.
1	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](#). If it is not implemented then this bit is RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 1.

LSMAOE, bit [4]**In ARMv8.3 and ARMv8.2:**

Load Multiple and Store Multiple Atomicity and Ordering Enable. When the OPTIONAL feature ARMv8.2-LSMAOC is implemented, defined values are:

LSMAOE	Meaning
0	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.
1	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.

If this bit is not implemented, it is RES1.

When this register has an architecturally-defined reset value, this field resets to 1.

In ARMv8.1 and ARMv8.0:

Reserved, RES1.

nTLSMD, bit [3]**In ARMv8.3 and ARMv8.2:**

No Trap Load Multiple and Store Multiple to Device-nGRE/Device-nGnRE/Device-nGnRnE memory. When the OPTIONAL feature ARMv8.2-LSMAOC is implemented, defined values are:

nTLSMD	Meaning
0	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.
1	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.

If this bit is not implemented, it is RES1.

When this register has an architecturally-defined reset value, this field resets to 1.

In ARMv8.1 and ARMv8.0:

Reserved, RES1.

C, bit [2]

Cacheability control, for data accesses at EL1 and EL0:

C	Meaning
0	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.
1	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 SCLTR.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.

When this register has an architecturally-defined reset value, 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
0	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.
1	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.

When this register has an architecturally-defined reset value, 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
0	EL1 and EL0 stage 1 address translation disabled. See the SCTLR.I field for the behavior of instruction accesses to Normal memory.
1	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}.

When this register has an architecturally-defined reset value, this field resets to 0.

Accessing the SCTLR

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c1, c0, 0	000	000	0001	1111	0000

Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	
EL3 not implemented	x	x	0	-	RW	n/a	n/a	SCTLR
EL3 not implemented	x	0	1	-	RW	RW	n/a	SCTLR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	SCTLR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	SCTLR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	SCTLR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	SCTLR

EL3 using AArch32	x	x	0	-	n/a	n/a	RW	SCTLR_s
EL3 using AArch32	x	0	1	-	RW	RW	RW	SCTLR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	SCTLR_ns

This table applies to all instructions that can access this register.

When EL3 is using AArch32, write access to SCTLR_s is UNDEFINED when the CP15SDISABLE signal is asserted HIGH.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.TVM](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR_EL2.TRVM](#)==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [HSTR_EL2.T1](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.TVM](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR_EL2.TRVM](#)==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [HSTR_EL2.T1](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HCR.TVM](#)==1, Non-secure write accesses to this register from EL1 are trapped to Hyp mode.
- If [HCR.TRVM](#)==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.
- If [HSTR.T1](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
(old)

htmldiff from-
SysReg v83A xml-00bet4

(new)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

VPIDR, Virtualization Processor ID Register

The VPIDR characteristics are:

Purpose

Holds the value of the Virtualization Processor ID. This is the value returned by Non-secure EL1 reads of [MIDR](#).

This register is part of:

- The Virtualization registers functional group.
- The Identification registers functional group.

Configuration

AArch32 System register VPIDR is architecturally mapped to AArch64 System register [VPIDR_EL2](#).

If EL2 is not implemented but EL3 is implemented, this register takes the value of the [MIDR](#).

Some or all RW fields of this register have defined reset values. These apply only if the PE resets into EL2 with EL2 using AArch32, or into EL3 with EL3 using AArch32. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

Attributes

VPIDR is a 32-bit register.

Field descriptions

The VPIDR bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Implementer								Variant				Architecture				PartNum								Revision							

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:

Hex representation	ASCII representation	Implementer
0x41	A	ARM Limited
0x42	B	Broadcom Corporation
0x43	C	Cavium Inc.
0x44	D	Digital Equipment Corporation
0x49	I	Infineon Technologies AG
0x4D	M	Motorola or Freescale Semiconductor Inc.
0x4E	N	NVIDIA Corporation
0x50	P	Applied Micro Circuits Corporation
0x51	Q	Qualcomm Inc.
0x56	V	Marvell International Ltd.
0x69	i	Intel Corporation

ARM can assign codes that are not published in this manual. All values not assigned by ARM are reserved and must not be used.

When this register has an architecturally-defined reset value, this field resets to the value of [MIDR](#).Implementer.

Variant, bits [23:20]

An IMPLEMENTATION DEFINED variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product.

When this register has an architecturally-defined reset value, this field resets to the value of [MIDR](#).Variant.

Architecture, bits [19:16]

The permitted values of this field are:

Architecture	Meaning
0001	ARMv4
0010	ARMv4T
0011	ARMv5 (obsolete)
0100	ARMv5T
0101	ARMv5TE
0110	ARMv5TEJ
0111	ARMv6
1111	Architectural features are individually identified in the ID_* registers, see 'ID Identification registers, functional group' in the ARMv8 ARM, section K12.5.3G4.18.1.

All other values are reserved.

When this register has an architecturally-defined reset value, this field resets to the value of [MIDR](#).Architecture.

PartNum, bits [15:4]

An IMPLEMENTATION DEFINED primary part number for the device.

On processors implemented by ARM, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently.

When this register has an architecturally-defined reset value, this field resets to the value of [MIDR](#).PartNum.

Revision, bits [3:0]

An IMPLEMENTATION DEFINED revision number for the device.

When this register has an architecturally-defined reset value, this field resets to the value of [MIDR](#).Revision.

Accessing the VPIDR

This register can be read using MRC with the following syntax:

```
MRC <syntax>
```

This register can be written using MCR with the following syntax:

```
MCR <syntax>
```

This syntax uses the following encoding in the System instruction encoding space:

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 4, <Rt>, c0, c0, 0	100	000	0000	1111	0000

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3

x	x	0	-	-	n/a	-
x	0	1	-	-	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile* for exceptions taken to AArch32 state, and section D1.13.2 (Synchronous exception prioritization) for exceptions taken to AArch64 state. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HSTR_EL2.T0==1](#), Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HSTR_EL2.T0==1](#), Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR_EL3.NS==1 :

- If [HSTR.T0==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg v83A xml-00bet5](#)

[SysReg v83A xml-00bet4](#)[\(old\)](#)

htmldiff from-

[\(new\)](#)[SysReg v83A xml-00bet4](#)[SysReg v83A xml-00bet5](#)

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

CP15DMB: Data Memory Barrier System instruction

CP15DSB: Data Synchronization Barrier System instruction

CP15ISB: Instruction Synchronization Barrier System instruction

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

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

TLBIMVAAAL: TLB Invalidate by VA, All ASID, Last level

TLBIMVAAALIS: 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

28/09/2017 08:46:41

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg v83A xml-00bet4
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
SysReg v83A xml-00bet5

[SysReg v83A xml-00bet4](#)

htmldiff from-

[\(new\)](#)[\(old\)](#)

SysReg_v83A_xml-00bet4

[SysReg v83A xml-00bet5](#)

AArch64 System Registers

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

AMAIR_EL1: Auxiliary Memory Attribute Indirection Register (EL1)

AMAIR_EL2: Auxiliary Memory Attribute Indirection Register (EL2)

AMAIR_EL3: Auxiliary Memory Attribute Indirection Register (EL3)

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])

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

CNTHP_CTL_EL2: Counter-timer Hypervisor Physical Timer Control register

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

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

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

CNTPCT_EL0: Counter-timer Physical Count 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

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

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)

ESR_EL1: Exception Syndrome Register (EL1)

ESR_EL2: Exception Syndrome Register (EL2)

ESR_EL3: Exception Syndrome Register (EL3)

ESR_ELx: Exception Syndrome Register (ELx)

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

HACR_EL2: Hypervisor Auxiliary Control Register

[HCR_EL2](#): Hypervisor Configuration 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_PMR_EL1: Interrupt Controller Interrupt Priority Mask Register

ICC_RPR_EL1: Interrupt Controller Running Priority Register

ICC_SGI0R_EL1: Interrupt Controller Software Generated Interrupt Group 0 Register

ICC_SGI1R_EL1: Interrupt Controller Software Generated Interrupt Group 1 Register

ICC_SRE_EL1: Interrupt Controller System Register Enable register (EL1)

ICC_SRE_EL2: Interrupt Controller System Register Enable register (EL2)

ICC_SRE_EL3: Interrupt Controller System Register Enable register (EL3)

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

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

ICH_EISR_EL2: Interrupt Controller End of Interrupt Status Register

ICH_ELRSR_EL2: Interrupt Controller Empty List Register Status Register

ICH_HCR_EL2: Interrupt Controller Hyp Control Register

ICH_LR<n>_EL2: Interrupt Controller List Registers

ICH_MISR_EL2: Interrupt Controller Maintenance Interrupt State Register

ICH_VMCR_EL2: Interrupt Controller Virtual Machine Control Register

ICH_VTR_EL2: Interrupt Controller VGIC Type Register

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

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

ICV_BPR0_EL1: Interrupt Controller Virtual Binary Point Register 0

ICV_BPR1_EL1: Interrupt Controller Virtual Binary Point Register 1

ICV_CTLR_EL1: Interrupt Controller Virtual Control Register

ICV_DIR_EL1: Interrupt Controller Deactivate Virtual Interrupt Register

ICV_EOIR0_EL1: Interrupt Controller Virtual End Of Interrupt Register 0

ICV_EOIR1_EL1: Interrupt Controller Virtual End Of Interrupt Register 1

ICV_HPPIR0_EL1: Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0

ICV_HPPIR1_EL1: Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1

ICV_IAR0_EL1: Interrupt Controller Virtual Interrupt Acknowledge Register 0

ICV_IAR1_EL1: Interrupt Controller Virtual Interrupt Acknowledge Register 1

ICV_IGRPEN0_EL1: Interrupt Controller Virtual Interrupt Group 0 Enable register

ICV_IGRPEN1_EL1: Interrupt Controller Virtual Interrupt Group 1 Enable register

ICV_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_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_AFR0_EL1: AArch32 Auxiliary Feature Register 0

ID_DFR0_EL1: AArch32 Debug Feature Register 0

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_PFR0_EL1: AArch32 Processor Feature Register 0

ID_PFR1_EL1: AArch32 Processor Feature Register 1

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

MDSR_EL1: Monitor Debug System Control Register

[MIDR_EL1](#): Main ID 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

PMOVSCLR_EL0: Performance Monitors Overflow Flag Status Clear Register

PMOVSSET_EL0: Performance Monitors Overflow Flag Status Set register

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

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

[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

PMSWINC_EL0: Performance Monitors Software Increment register

PMUSERENR_EL0: Performance Monitors User Enable Register

PMXVCNTR_EL0: Performance Monitors Selected Event Count Register

PMXEVTPER_EL0: Performance Monitors Selected Event Type Register

REVIDR_EL1: Revision ID Register

RMR_EL1: Reset Management Register (EL1)

RMR_EL2: Reset Management Register (EL2)

RMR_EL3: Reset Management Register (EL3)

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)

SDER32_EL3: AArch32 Secure Debug Enable Register

SPSR_EL1: Saved Program Status Register (EL1)

SPSR_EL2: Saved Program Status Register (EL2)

SPSR_EL3: Saved Program Status Register (EL3)

SPSR_abt: Saved Program Status Register (Abort mode)

SPSR_fiq: Saved Program Status Register (FIQ mode)

SPSR_irq: Saved Program Status Register (IRQ mode)

SPSR_und: Saved Program Status Register (Undefined mode)

SPSel: Stack Pointer Select

SP_EL0: Stack Pointer (EL0)

SP_EL1: Stack Pointer (EL1)

SP_EL2: Stack Pointer (EL2)

SP_EL3: Stack Pointer (EL3)

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

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

TCR_EL3: Translation Control Register (EL3)

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

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)

VMPIDR_EL2: Virtualization Multiprocessor ID Register

[VPIDR_EL2](#): Virtualization Processor ID Register

VTCR_EL2: Virtualization Translation Control Register

VTTBR_EL2: Virtualization Translation Table Base Register

28/09/2017 08:16:41

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTHP_CVAL_EL2, Counter-timer Hypervisor Physical Timer CompareValue register

The CNTHP_CVAL_EL2 characteristics are:

Purpose

Holds the compare value for the EL2 physical timer.

This register is part of:

- The Generic Timer registers functional group.
- The Virtualization registers functional group.

Configuration

AArch64 System register CNTHP_CVAL_EL2 is architecturally mapped to AArch32 System register [CNTHP_CVAL](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTHP_CVAL_EL2 is a 64-bit register.

Field descriptions

The CNTHP_CVAL_EL2 bit assignments are:

63	62	61	60	59	58	57	56	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 ([CNTPCT_EL0](#) - CompareValue) is greater than or equal to zero. This means that CompareValue acts like a 64-bit upcounter timer. When the timer condition is met:

- [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.

Accessing the CNTHP_CVAL_EL2

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

MSR <systemreg>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
CNTHP_CVAL_EL2	11	100	1110	0010	010
CNTP_CVAL_EL0	11	011	1110	0010	010

Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTHP_CVAL_EL2	x	x	0	-	-	n/a	RW
CNTHP_CVAL_EL2	0	0	1	-	-	RW	RW
CNTHP_CVAL_EL2	0	1	1	-	n/a	RW	RW
CNTHP_CVAL_EL2	1	0	1	-	-	RW	RW
CNTHP_CVAL_EL2	1	1	1	-	n/a	RW	RW
CNTP_CVAL_EL0	x	x	0	CNTP_CVAL_EL0	CNTP_CVAL_EL0	n/a	CNTP_CVAL_EL0
CNTP_CVAL_EL0	0	0	1	CNTP_CVAL_EL0	CNTP_CVAL_EL0	CNTP_CVAL_EL0	CNTP_CVAL_EL0
CNTP_CVAL_EL0	0	1	1	CNTP_CVAL_EL0	n/a	CNTP_CVAL_EL0	CNTP_CVAL_EL0
CNTP_CVAL_EL0	1	0	1	CNTP_CVAL_EL0	CNTP_CVAL_EL0	RW	RW
CNTP_CVAL_EL0	1	1	1	RW	n/a	RW	RW

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.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2.EL0PTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTHP_TVAL_EL2, Counter-timer Hypervisor Physical Timer TimerValue register

The CNTHP_TVAL_EL2 characteristics are:

Purpose

Holds the timer value for the EL2 physical timer.

This register is part of:

- The Generic Timer registers functional group.
- The Virtualization registers functional group.

Configuration

AArch64 System register CNTHP_TVAL_EL2 is architecturally mapped to AArch32 System register [CNTHP_TVAL](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTHP_TVAL_EL2 is a 32-bit register.

Field descriptions

The CNTHP_TVAL_EL2 bit assignments are:

31	30	29	28	27	26	25	24	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_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.

Accessing the CNTHP_TVAL_EL2

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
CNTHP_TVAL_EL2	11	100	1110	0010	000
CNTP_TVAL_EL0	11	011	1110	0010	000

Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTHP_TVAL_EL2	x	x	0	-	-	n/a	RW
CNTHP_TVAL_EL2	0	0	1	-	-	RW	RW
CNTHP_TVAL_EL2	0	1	1	-	n/a	RW	RW
CNTHP_TVAL_EL2	1	0	1	-	-	RW	RW
CNTHP_TVAL_EL2	1	1	1	-	n/a	RW	RW
CNTP_TVAL_EL0	x	x	0	CNTP_TVAL_EL0	CNTP_TVAL_EL0	n/a	CNTP_TVAL_EL0
CNTP_TVAL_EL0	0	0	1	CNTP_TVAL_EL0	CNTP_TVAL_EL0	CNTP_TVAL_EL0	CNTP_TVAL_EL0
CNTP_TVAL_EL0	0	1	1	CNTP_TVAL_EL0	n/a	CNTP_TVAL_EL0	CNTP_TVAL_EL0
CNTP_TVAL_EL0	1	0	1	CNTP_TVAL_EL0	CNTP_TVAL_EL0	RW	RW
CNTP_TVAL_EL0	1	1	1	RW	n/a	RW	RW

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.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2.ELOPTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

<u>SysReg_v83A_xml-00bet4</u> <u>(old)</u>	htmldiff from- SysReg_v83A_xml-00bet4	<u>(new)</u> <u>SysReg_v83A_xml-00bet5</u>
---	--	---

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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.

This register is part of the Generic Timer registers functional group.

Configuration

AArch64 System register CNTHV_CVAL_EL2 is architecturally mapped to AArch32 System register [CNTHV_CVAL](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

This register is introduced in ARMv8.1.

Attributes

CNTHV_CVAL_EL2 is a 64-bit register.

Field descriptions

The CNTHV_CVAL_EL2 bit assignments are:

63	62	61	60	59	58	57	56	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 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.

Accessing the CNTHV_CVAL_EL2

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
CNTHV_CVAL_EL2	11	100	1110	0011	010
CNTV_CVAL_EL0	11	011	1110	0011	010

Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTHV_CVAL_EL2	x	x	0	-	-	n/a	RW
CNTHV_CVAL_EL2	0	0	1	-	-	RW	RW
CNTHV_CVAL_EL2	0	1	1	-	n/a	RW	RW
CNTHV_CVAL_EL2	1	0	1	-	-	RW	RW
CNTHV_CVAL_EL2	1	1	1	-	n/a	RW	RW
CNTV_CVAL_EL0	x	x	0	CNTV_CVAL_EL0	CNTV_CVAL_EL0	n/a	CNTV_CVAL_EL0
CNTV_CVAL_EL0	0	0	1	CNTV_CVAL_EL0	CNTV_CVAL_EL0	CNTV_CVAL_EL0	CNTV_CVAL_EL0
CNTV_CVAL_EL0	0	1	1	CNTV_CVAL_EL0	n/a	CNTV_CVAL_EL0	CNTV_CVAL_EL0
CNTV_CVAL_EL0	1	0	1	CNTV_CVAL_EL0	CNTV_CVAL_EL0	RW	CNTV_CVAL_EL0
CNTV_CVAL_EL0	1	1	1	RW	n/a	RW	CNTV_CVAL_EL0

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.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2.EL0VTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

<u>SysReg_v83A_xml-00bet4</u> <u>(old)</u>	htmldiff from- SysReg_v83A_xml-00bet4	<u>(new)</u> <u>SysReg_v83A_xml-00bet5</u>
---	--	---

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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.

This register is part of the Generic Timer registers functional group.

Configuration

AArch64 System register CNTHV_TVAL_EL2 is architecturally mapped to AArch32 System register [CNTHV_TVAL](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

This register is introduced in ARMv8.1.

Attributes

CNTHV_TVAL_EL2 is a 32-bit register.

Field descriptions

The CNTHV_TVAL_EL2 bit assignments are:

31	30	29	28	27	26	25	24	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_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.

Accessing the CNTHV_TVAL_EL2

This register can be read using MRS with the following syntax:

MRS <Xt>, <systemreg>

This register can be written using MSR (register) with the following syntax:

MSR <systemreg>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
CNTHV_TVAL_EL2	11	100	1110	0011	000
CNTV_TVAL_EL0	11	011	1110	0011	000

Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTHV_TVAL_EL2	x	x	0	-	-	n/a	RW
CNTHV_TVAL_EL2	0	0	1	-	-	RW	RW
CNTHV_TVAL_EL2	0	1	1	-	n/a	RW	RW
CNTHV_TVAL_EL2	1	0	1	-	-	RW	RW
CNTHV_TVAL_EL2	1	1	1	-	n/a	RW	RW
CNTV_TVAL_EL0	x	x	0	CNTV_TVAL_EL0	CNTV_TVAL_EL0	n/a	CNTV_TVAL_EL0
CNTV_TVAL_EL0	0	0	1	CNTV_TVAL_EL0	CNTV_TVAL_EL0	CNTV_TVAL_EL0	CNTV_TVAL_EL0
CNTV_TVAL_EL0	0	1	1	CNTV_TVAL_EL0	n/a	CNTV_TVAL_EL0	CNTV_TVAL_EL0
CNTV_TVAL_EL0	1	0	1	CNTV_TVAL_EL0	CNTV_TVAL_EL0	RW	CNTV_TVAL_EL0
CNTV_TVAL_EL0	1	1	1	RW	n/a	RW	CNTV_TVAL_EL0

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.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2.EL0VTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTPS_CVAL_EL1, Counter-timer Physical Secure Timer CompareValue register

The CNTPS_CVAL_EL1 characteristics are:

Purpose

Holds the compare value for the secure physical timer, usually accessible at EL3 but configurably accessible at EL1 in Secure state.

This register is part of the Generic Timer registers functional group.

Configuration

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTPS_CVAL_EL1 is a 64-bit register.

Field descriptions

The CNTPS_CVAL_EL1 bit assignments are:

63	62	61	60	59	58	57	56	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 secure physical timer CompareValue.

When [CNTPS_CTL_EL1](#).ENABLE is 1, the timer condition is met when ([CNTPCT_EL0](#) - CompareValue) is greater than or equal to zero. This means that CompareValue acts like a 64-bit upcounter timer. When the timer condition is met:

- [CNTPS_CTL_EL1](#).ISTATUS is set to 1.
- If [CNTPS_CTL_EL1](#).IMASK is 0, an interrupt is generated.

When [CNTPS_CTL_EL1](#).ENABLE is 0, the timer condition is not met, but [CNTPCT_EL0](#) continues to count.

Accessing the CNTPS_CVAL_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
CNTPS_CVAL_EL1	11	111	1110	0010	010

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
0	0	1	-	-	-	RW
0	1	1	-	n/a	-	RW
1	0	1	-	-	-	RW
1	1	1	-	n/a	-	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL3 is implemented and is using AArch64 and SCR_EL3.NS==0 :

- If [SCR_EL3.ST](#)=0, Secure accesses to this register from EL1 are trapped to EL3.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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.

This register is part of the Generic Timer registers functional group.

Configuration

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTPS_TVAL_EL1 is a 32-bit register.

Field descriptions

The CNTPS_TVAL_EL1 bit assignments are:

31	30	29	28	27	26	25	24	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 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.

Accessing the CNTPS_TVAL_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
CNTPS_TVAL_EL1	11	111	1110	0010	000

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
0	0	1	-	-	-	RW
0	1	1	-	n/a	-	RW
1	0	1	-	-	-	RW
1	1	1	-	n/a	-	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL3 is implemented and is using AArch64 and SCR_EL3.NS==0 :

- If [SCR_EL3.ST==0](#), Secure accesses to this register from EL1 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTP_CVAL_EL0, Counter-timer Physical Timer CompareValue register

The CNTP_CVAL_EL0 characteristics are:

Purpose

Holds the compare value for the EL1 physical timer.

This register is part of the Generic Timer registers functional group.

Configuration

AArch64 System register CNTP_CVAL_EL0 is architecturally mapped to AArch32 System register [CNTP_CVAL](#).

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTP_CVAL_EL0 is a 64-bit register.

Field descriptions

The CNTP_CVAL_EL0 bit assignments are:

63	62	61	60	59	58	57	56	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 EL1 physical timer CompareValue.

When [CNTP_CTL_EL0](#).ENABLE is 1, the timer condition is met when ([CNTPCT_EL0](#) - CompareValue) is greater than or equal to zero. This means that CompareValue acts like a 64-bit upcounter timer. When the timer condition is met:

- [CNTP_CTL_EL0](#).ISTATUS is set to 1.
- If [CNTP_CTL_EL0](#).IMASK is 0, an interrupt is generated.

When [CNTP_CTL_EL0](#).ENABLE is 0, the timer condition is not met, but [CNTPCT_EL0](#) continues to count.

Accessing the CNTP_CVAL_EL0

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
-------------	-----	-----	-----	-----	-----

CNTP_CVAL_EL0	11	011	1110	0010	010
CNTP_CVAL_EL02	11	101	1110	0010	010

Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTP_CVAL_EL0	x	x	0	RW	RW	n/a	RW
CNTP_CVAL_EL0	0	0	1	RW	RW	RW	RW
CNTP_CVAL_EL0	0	1	1	RW	n/a	RW	RW
CNTP_CVAL_EL0	1	0	1	RW	RW	CNTHP_CVAL_EL2	RW
CNTP_CVAL_EL0	1	1	1	CNTHP_CVAL_EL2	n/a	CNTHP_CVAL_EL2	RW
CNTP_CVAL_EL02	x	x	0	-	-	n/a	-
CNTP_CVAL_EL02	0	0	1	-	-	-	-
CNTP_CVAL_EL02	0	1	1	-	n/a	-	-
CNTP_CVAL_EL02	1	0	1	-	-	RW	RW
CNTP_CVAL_EL02	1	1	1	-	n/a	RW	RW

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic CNTP_CVAL_EL0 or CNTP_CVAL_EL02 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When HCR_EL2.E2H==0 :

- If [CNTKCTL_EL1.EL0PTEN](#)==0, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [CNTHCTL_EL2.EL1PCEN](#)==0, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [CNTHCTL_EL2.EL1PCEN](#)==0, and [CNTKCTL_EL1.EL0PTEN](#)==1, Non-secure accesses to this register from EL0 are trapped to EL2.
- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 using accessor CNTP_CVAL_EL02 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [CNTHCTL_EL2.EL1PTEN](#)==0, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [CNTHCTL_EL2.EL1PTEN](#)==0, and [CNTKCTL_EL1.EL0PTEN](#)==1, Non-secure accesses to this register from EL0 are trapped to EL2.
- If [CNTKCTL_EL1.EL0PTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL1.
- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 using accessor CNTP_CVAL_EL02 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2.EL0PTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

<u>SysReg_v83A_xml-00bet4</u> <u>(old)</u>	htmldiff from- SysReg_v83A_xml-00bet4	<u>(new)</u> <u>SysReg_v83A_xml-00bet5</u>
---	--	---

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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.

This register is part of the Generic Timer registers functional group.

Configuration

AArch64 System register CNTP_TVAL_ELO is architecturally mapped to AArch32 System register [CNTP_TVAL](#).

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTP_TVAL_ELO is a 32-bit register.

Field descriptions

The CNTP_TVAL_ELO bit assignments are:

31	30	29	28	27	26	25	24	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_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.

Accessing the CNTP_TVAL_ELO

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

MSR <systemreg>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
CNTP_TVAL_EL0	11	011	1110	0010	000
CNTP_TVAL_EL02	11	101	1110	0010	000

Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTP_TVAL_EL0	x	x	0	RW	RW	n/a	RW
CNTP_TVAL_EL0	0	0	1	RW	RW	RW	RW
CNTP_TVAL_EL0	0	1	1	RW	n/a	RW	RW
CNTP_TVAL_EL0	1	0	1	RW	RW	CNTHP_TVAL_EL2	RW
CNTP_TVAL_EL0	1	1	1	CNTHP_TVAL_EL2	n/a	CNTHP_TVAL_EL2	RW
CNTP_TVAL_EL02	x	x	0	-	-	n/a	-
CNTP_TVAL_EL02	0	0	1	-	-	-	-
CNTP_TVAL_EL02	0	1	1	-	n/a	-	-
CNTP_TVAL_EL02	1	0	1	-	-	RW	RW
CNTP_TVAL_EL02	1	1	1	-	n/a	RW	RW

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.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When HCR_EL2.E2H==0 :

- If [CNTKCTL_EL1.EL0PTEN](#)==0, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [CNTHCTL_EL2.EL1PCEN](#)==0, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [CNTHCTL_EL2.EL1PCEN](#)==0, and [CNTKCTL_EL1.EL0PTEN](#)==1, Non-secure accesses to this register from EL0 are trapped to EL2.
- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 using accessor CNTP_TVAL_EL02 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [CNTHCTL_EL2.EL1PTEN](#)==0, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [CNTHCTL_EL2.EL1PTEN](#)==0, and [CNTKCTL_EL1.EL0PTEN](#)==1, Non-secure accesses to this register from EL0 are trapped to EL2.
- If [CNTKCTL_EL1.EL0PTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL1.
- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 using accessor CNTP_TVAL_EL02 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2.EL0PTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

<u>SysReg v83A xml-00bet4</u> <u>(old)</u>	htmldiff from-	<u>(new)</u>
	SysReg_v83A_xml-00bet4	<u>SysReg v83A xml-00bet5</u>

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTV_CVAL_EL0, Counter-timer Virtual Timer CompareValue register

The CNTV_CVAL_EL0 characteristics are:

Purpose

Holds the compare value for the virtual timer.

This register is part of the Generic Timer registers functional group.

Configuration

AArch64 System register CNTV_CVAL_EL0 is architecturally mapped to AArch32 System register [CNTV_CVAL](#).

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTV_CVAL_EL0 is a 64-bit register.

Field descriptions

The CNTV_CVAL_EL0 bit assignments are:

63	62	61	60	59	58	57	56	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 EL1 virtual timer CompareValue.

When [CNTV_CTL_EL0.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:

- [CNTV_CTL_EL0.ISTATUS](#) is set to 1.
- If [CNTV_CTL_EL0.IMASK](#) is 0, an interrupt is generated.

When [CNTV_CTL_EL0.ENABLE](#) is 0, the timer condition is not met, but [CNTVCT_EL0](#) continues to count.

Accessing the CNTV_CVAL_EL0

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
-------------	-----	-----	-----	-----	-----

CNTV_CVAL_EL0	11	011	1110	0011	010
CNTV_CVAL_EL02	11	101	1110	0011	010

Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTV_CVAL_EL0	x	x	0	RW	RW	n/a	RW
CNTV_CVAL_EL0	0	0	1	RW	RW	RW	RW
CNTV_CVAL_EL0	0	1	1	RW	n/a	RW	RW
CNTV_CVAL_EL0	1	0	1	RW	RW	CNTHV_CVAL_EL2	RW
CNTV_CVAL_EL0	1	1	1	CNTHV_CVAL_EL2	n/a	CNTHV_CVAL_EL2	RW
CNTV_CVAL_EL02	x	x	0	-	-	n/a	-
CNTV_CVAL_EL02	0	0	1	-	-	-	-
CNTV_CVAL_EL02	0	1	1	-	n/a	-	-
CNTV_CVAL_EL02	1	0	1	-	-	RW	RW
CNTV_CVAL_EL02	1	1	1	-	n/a	RW	RW

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic CNTV_CVAL_EL0 or CNTV_CVAL_EL02 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When [HCR_EL2.E2H](#)==0 :

- If [CNTKCTL_EL1.EL0VTEN](#)==0, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 using accessor CNTV_CVAL_EL02 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==1 && [HCR_EL2.TGE](#)==0 :

- If [CNTKCTL_EL1.EL0VTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL1.
- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 using accessor CNTV_CVAL_EL02 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 && [HCR_EL2.E2H](#)==1 && [HCR_EL2.TGE](#)==1 :

- If [CNTHCTL_EL2.EL0VTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTV_TVAL_EL0, Counter-timer Virtual Timer TimerValue register

The CNTV_TVAL_EL0 characteristics are:

Purpose

Holds the timer value for the EL1 virtual timer.

This register is part of the Generic Timer registers functional group.

Configuration

AArch64 System register CNTV_TVAL_EL0 is architecturally mapped to AArch32 System register [CNTV_TVAL](#).

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CNTV_TVAL_EL0 is a 32-bit register.

Field descriptions

The CNTV_TVAL_EL0 bit assignments are:

31	30	29	28	27	26	25	24	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 virtual timer.

On a read of this register:

- If [CNTV_CTL_EL0.ENABLE](#) is 0, the value returned is UNKNOWN.
- If [CNTV_CTL_EL0.ENABLE](#) is 1, the value returned is $(\text{CNTV_CVAL_EL0} - (\text{CNTVOFF_EL2} \text{CNTVCT_EL0} \text{CNTPCT_EL0}))$.

On a write of this register, [CNTV_CVAL_EL0](#) is set to $(\text{CNTVCT_EL0} \text{CNTPCT_EL0} + \text{TimerValue})$, where TimerValue is treated as a signed 32-bit integer.

When [CNTV_CTL_EL0.ENABLE](#) is 1, the timer condition is met when $(\text{CNTVCT_EL0} \text{CNTPCT_EL0} - \text{CNTV_CVAL_EL0})$ 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_EL0.ISTATUS](#) is set to 1.
- If [CNTV_CTL_EL0.IMASK](#) is 0, an interrupt is generated.

When [CNTV_CTL_EL0.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.

Accessing the CNTV_TVAL_EL0

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
CNTV_TVAL_EL0	11	011	1110	0011	000
CNTV_TVAL_EL02	11	101	1110	0011	000

Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTV_TVAL_EL0	x	x	0	RW	RW	n/a	RW
CNTV_TVAL_EL0	0	0	1	RW	RW	RW	RW
CNTV_TVAL_EL0	0	1	1	RW	n/a	RW	RW
CNTV_TVAL_EL0	1	0	1	RW	RW	CNTHV_TVAL_EL2	RW
CNTV_TVAL_EL0	1	1	1	CNTHV_TVAL_EL2	n/a	CNTHV_TVAL_EL2	RW
CNTV_TVAL_EL02	x	x	0	-	-	n/a	-
CNTV_TVAL_EL02	0	0	1	-	-	-	-
CNTV_TVAL_EL02	0	1	1	-	n/a	-	-
CNTV_TVAL_EL02	1	0	1	-	-	RW	RW
CNTV_TVAL_EL02	1	1	1	-	n/a	RW	RW

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.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When HCR_EL2.E2H==0 :

- If [CNTKCTL_EL1.EL0VTEN](#)==0, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 using accessor CNTV_TVAL_EL02 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [CNTKCTL_EL1.EL0VTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL1.
- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 using accessor CNTV_TVAL_EL02 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==1 :

- If [CNTHCTL_EL2.EL0VTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

<u>SysReg v83A xml-00bet4</u> <u>(old)</u>	htmldiff from-	<u>(new)</u>
	SysReg_v83A_xml-00bet4	<u>SysReg v83A xml-00bet5</u>

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CSSELR_EL1, Cache Size Selection Register

The CSSELR_EL1 characteristics are:

Purpose

Selects the current Cache Size ID Register, [CCSIDR_EL1](#), by specifying the required cache level and the cache type (either instruction or data cache).

This register is part of the Identification registers functional group.

Configuration

AArch64 System register CSSELR_EL1 is architecturally mapped to AArch32 System register [CSSELR](#).

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

CSSELR_EL1 is a 32-bit register.

Field descriptions

The CSSELR_EL1 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Level	InD	

Bits [31:4]

Reserved, RES0.

Level, bits [3:1]

Cache level of required cache. Permitted values are:

Level	Meaning
000	Level 1 cache
001	Level 2 cache
010	Level 3 cache
011	Level 4 cache
100	Level 5 cache
101	Level 6 cache
110	Level 7 cache

All other values are reserved.

If CSSELR_EL1.Level is programmed to a cache level that is not implemented, then the value for this field on a read of CSSELR is UNKNOWN.

InD, bit [0]

Instruction not Data bit. Permitted values are:

InD	Meaning
0	Data or unified cache.
1	Instruction cache.

If CSSELR_EL1.Level is programmed to a cache level that is not implemented, then the value for this field on a read of CSSELR is UNKNOWN.

Accessing the CSSELR_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
CSSELR_EL1	11	010	0000	0000	000

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
x	0	1	-	RW	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.TID2](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.TID2](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

DBGBCR<n>_EL1, Debug Breakpoint Control Registers, n = 0 - 15

The DBGBCR<n>_EL1 characteristics are:

Purpose

Holds control information for a breakpoint. Forms breakpoint n together with value register [DBGBVR<n>_EL1](#).

This register is part of the Debug registers functional group.

Configuration

AArch64 System register DBGBCR<n>_EL1 is architecturally mapped to AArch32 System register [DBGBCR<n>](#).

AArch64 System register DBGBCR<n>_EL1 is architecturally mapped to External register [DBGBCR<n>_EL1](#).

If breakpoint n is not implemented then this register is unallocated.

This register is in the Cold reset domain. On a Cold reset RW fields in this register reset to architecturally UNKNOWN values. The register is not affected by a Warm reset.

Attributes

DBGBCR<n>_EL1 is a 32-bit register.

Field descriptions

The DBGBCR<n>_EL1 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0		BT				LBN			SSC	HMC	0	0	0	0		BAS			0	0	PMC	E		

Bits [31:24]

Reserved, RES0.

BT, bits [23:20]

Breakpoint Type. Possible values are:

BT	Meaning
0000	Unlinked instruction address match.
0001	Linked instruction address match.
0010	Unlinked Context ID match.
0011	Linked Context ID match.
0110	Unlinked CONTEXTIDR_EL1 match (introduced in ARMv8.1).
0111	Linked CONTEXTIDR_EL1 match (introduced in ARMv8.1).
1000	Unlinked VMID match.
1001	Linked VMID match.
1010	Unlinked VMID and Context ID match.
1011	Linked VMID and Context ID match.
1100	Unlinked CONTEXTIDR_EL2 match (introduced in ARMv8.1).
1101	Linked CONTEXTIDR_EL2 match (introduced in ARMv8.1).
1110	Unlinked Full Context ID match (introduced in ARMv8.1).
1111	Linked Full Context ID match (introduced in ARMv8.1).

The field breaks down as follows:

- BT[3:1]: Base type.
 - 000**
Match address. [DBGBVR<n>_EL1](#) is the address of an instruction.
 - 001**
Match Context ID. [DBGBVR<n>_EL1](#).ContextID is a Context ID compared against [CONTEXTIDR_EL1](#) when ARMv8.1-VHE is not implemented, or not in a Host OS or a Host Application. When ARMv8.1-VHE is implemented, and in a Host OS or Host Application, the Context ID is compared against [CONTEXTIDR_EL2](#).
 - 011**
Match [CONTEXTIDR_EL1](#). [DBGBVR<n>_EL1](#).ContextID is a Context ID compared against [CONTEXTIDR_EL1](#).
 - 100**
Match VMID. [DBGBVR<n>_EL1](#).VMID is a VMID compared against [VTTBR_EL2](#).VMID.
 - 101**
Match VMID and Context ID. [DBGBVR<n>_EL1](#).ContextID is a Context ID compared against [CONTEXTIDR_EL1](#), and [DBGBVR<n>_EL1](#).VMID is a VMID compared against [VTTBR_EL2](#).VMID.
 - 110**
Match [CONTEXTIDR_EL2](#). [DBGBVR<n>_EL1](#).ContextID2 is a Context ID compared against [CONTEXTIDR_EL2](#).
 - 111**
Match Full Context ID. [DBGBVR<n>_EL1](#).ContextID is compared against [CONTEXTIDR_EL1](#), and [DBGBVR<n>_EL1](#).ContextID2 is compared against [CONTEXTIDR_EL2](#).
- BT[0]: Enable linking.

All other values are reserved. Constraints on breakpoint programming mean other values are reserved under some conditions. For more information, including the effect of programming this field to a reserved value, see 'Reserved DBGBCR<n>_EL1.BT values' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

LBN, bits [19:16]

Linked breakpoint number. For Linked address matching breakpoints, this specifies the index of the Context-matching breakpoint linked to.

For all other breakpoint types this field is ignored and reads of the register return an UNKNOWN value.

This field is ignored when the value of DBGBCR<n>_EL1.E is 0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

SSC, bits [15:14]

Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the HMC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information, including the effect of programming the fields to a reserved set of values, see 'Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).

For more information on the operation of the SSC, HMC, and PMC fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

HMC, bit [13]

Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and PMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the SSC, bits [15:14] description.

For more information on the operation of the SSC, HMC, and PMC fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bits [12:9]

Reserved, RES0.

BAS, bits [8:5]

Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state. In an AArch64-only implementation, this field is reserved, RES1.

The permitted values depend on the breakpoint type.

For Address match breakpoints, the permitted values are:

BAS	Match instruction at	Constraint for debuggers
0011	DBGBVR<n>_EL1	Use for T32 instructions.
1100	DBGBVR<n>_EL1+2	Use for T32 instructions.
1111	DBGBVR<n>_EL1	Use for A64 and A32 instructions.

All other values are reserved. For more information, see 'Reserved DBGBCR<n>_EL1.BAS values' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).

For more information on using the BAS field in address match breakpoints, see 'Using the BAS field in Address Match breakpoints' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug).

For Context matching breakpoints, this field is RES1 and ignored.

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.

Bits [4:3]

Reserved, RES0.

PMC, bits [2:1]

Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated. This field must be interpreted along with the SSC and HMC fields, and there are constraints on the permitted values of the {HMC, SSC, PMC} fields. For more information see the DBGBCR<n>_EL1.SSC description.

For more information on the operation of the SSC, HMC, and PMC fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

E, bit [0]

Enable breakpoint [DBGBVR<n>_EL1](#). Possible values are:

E	Meaning
0	Breakpoint disabled.
1	Breakpoint enabled.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Accessing the DBGBCR<n>_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
DBGBCR<n>_EL1	10	000	0000	n<3:0>	101

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
x	0	1	-	RW	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [EDSCR.TDA](#)==1, and [OSLSR_EL1.OSLK](#)==0, and halting is allowed, accesses to this register from EL1, EL2, and EL3 generate a trapped Software Access Debug event state.

When EL2 is implemented and is using AArch64 and [SCR_EL3.NS](#)==1 :

- If [MDCR_EL2.TDA](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3.TDA](#)==1, accesses to this register from EL1 and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

DBGBVR<n>_EL1, Debug Breakpoint Value Registers, n = 0 - 15

The DBGBVR<n>_EL1 characteristics are:

Purpose

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register [DBGBCR<n>_EL1](#).

This register is part of the Debug registers functional group.

Configuration

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

AArch64 System register DBGBVR<n>_EL1 bits [63:32] are architecturally mapped to AArch32 System register [DBGXVR<n>](#).

AArch64 System register DBGBVR<n>_EL1 is architecturally mapped to External register [DBGBVR<n>_EL1](#).

If breakpoint n is not implemented then this register is unallocated.

This register is in the Cold reset domain. On a Cold reset RW fields in this register reset to architecturally UNKNOWN values. The register is not affected by a Warm reset.

Attributes

How this register is interpreted depends on the value of [DBGBCR<n>_EL1](#).BT.

- When [DBGBCR<n>_EL1](#).BT is 0b000x, this register holds a virtual address.
- When [DBGBCR<n>_EL1](#).BT is 0b001x, 0b011x, or 0b110x, this register holds a Context ID.
- When [DBGBCR<n>_EL1](#).BT is 0b100x, this register holds a VMID.
- When [DBGBCR<n>_EL1](#).BT is 0b101x, this register holds a VMID and a Context ID.
- When [DBGBCR<n>_EL1](#).BT is 0b111x, this register holds two Context ID values.

For other values of [DBGBCR<n>_EL1](#).BT, this register is RES0.

Field descriptions

The DBGBVR<n>_EL1 bit assignments are:

When DBGBCR<n>_EL1.BT==0b000x:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32			
RESS[14:4]											VA[52:49]					VA[48:2]																		
VA[48:2]																															0		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			

RESS[14:4], bits [63:53]

Reserved, Sign extended. Software must treat this field as RES0 if the most significant bit of VA is 0 or RES0, and as RES1 if the most significant bit of VA is 1.

Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether:

- The bits are hardwired to a copy of the most significant bit of VA, meaning writes to these bits are ignored, and reads to the bits always return the hardwired value.
- The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.

VA[52:49], bits [52:49]

In ARMv8.3 and ARMv8.2:

Extension to VA[48:2]. See VA[48:2] for more details.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.1 and ARMv8.0:

Extension to RESS[14:4]. See RESS[14:4] for more details.

VA[48:2], bits [48:2]

Bits[48:2] of the address value for comparison.

When ARMv8.2-LVA is implemented, and 52-bit addresses and a 64KB translation granule are in use, VA[52:49] form the upper part of the address value. Otherwise, VA[52:49] are RESS.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bits [1:0]

Reserved, RES0.

When DBGBCR<n>_EL1.BT==0b001x:

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

Bits [63:32]

Reserved, RES0.

ContextID, bits [31:0]

Context ID value for comparison.

The value is compared against [CONTEXTIDR_EL1](#) in the following cases:

- The PE is in Secure state.
- When ARMv8.1-VHE is not implemented.
- When ARMv8.1-VHE is implemented, [HCR_EL2.E2H](#) is 0 and the PE is in Non-secure EL0, EL1 or EL2.
- When ARMv8.1-VHE is implemented, [HCR_EL2](#).{E2H, TGE} is {1, 0} and the PE is in Non-secure EL0 or EL1.

When ARMv8.1-VHE is implemented, [HCR_EL2.E2H](#) is 1, the value is compared against [CONTEXTIDR_EL2](#) in the following cases:

- The PE is executing at EL2.
- [HCR_EL2.TGE](#) is 1 and the PE is in Non-secure EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

When DBGBCR<n>_EL1.BT==0b011x:

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

Bits [63:32]

Reserved, RES0.

ContextID, bits [31:0]

Context ID value for comparison against [CONTEXTIDR_EL1](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

When DBGBCR<n>_EL1.BT==0b100x and EL2 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VMID[15:8]								VMID[7:0]							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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 [63:48]

Reserved, RES0.

VMID[15:8], bits [47:40]

In ARMv8.3, ARMv8.2 and ARMv8.1:

Extension to VMID[7:0]. See VMID[7:0] for more details.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.0:

Reserved, RES0.

VMID[7:0], bits [39:32]

VMID value for comparison.

The VMID is 8 bits in the following cases.

- EL2 is using AArch32.
- ARMv8.1-VMID16 is not implemented.

When ARMv8.1-VMID16 is implemented and EL2 is using AArch64, it is IMPLEMENTATION DEFINED whether the VMID is 8 bits or 16 bits.

VMID[15:8] is RES0 if any of the following applies:

- The implementation has an 8-bit VMID.
- [VTCR_EL2](#).VS has a value of 0.
- EL2 is using AArch32.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bits [31:0]

Reserved, RES0.

When DBGBCR<n>_EL1.BT==0b101x and EL2 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VMID[15:8]								VMID[7:0]							
ContextID																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:48]

Reserved, RES0.

VMID[15:8], bits [47:40]

In ARMv8.3, ARMv8.2 and ARMv8.1:

Extension to VMID[7:0]. See VMID[7:0] for more details.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.0:

Reserved, RES0.

VMID[7:0], bits [39:32]

VMID value for comparison.

The VMID is 8 bits in the following cases.

- EL2 is using AArch32.
- ARMv8.1-VMID16 is not implemented.

When ARMv8.1-VMID16 is implemented and EL2 is using AArch64, it is IMPLEMENTATION DEFINED whether the VMID is 8 bits or 16 bits.

VMID[15:8] is RES0 if any of the following applies:

- The implementation has an 8-bit VMID.
- [VTCR_EL2](#).VS has a value of 0.
- EL2 is using AArch32.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

ContextID, bits [31:0]

Context ID value for comparison against [CONTEXTIDR_EL1](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

When DBGBCR<n>_EL1.BT==0b110x and EL2 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
ContextID2																															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

ContextID2, bits [63:32]

Context ID value for comparison against [CONTEXTIDR_EL2](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bits [31:0]

Reserved, RES0.

When DBGBCR<n>_EL1.BT==0b111x and EL2 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
ContextID2																															
ContextID																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

ContextID2, bits [63:32]

Context ID value for comparison against [CONTEXTIDR_EL2](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

ContextID, bits [31:0]

Context ID value for comparison against [CONTEXTIDR_EL1](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Accessing the DBGBVR<n>_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
DBGBVR<n>_EL1	10	000	0000	n<3:0>	100

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
x	0	1	-	RW	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [EDSCR.TDA](#)==1, and [OSLSR_EL1.OSLK](#)==0, and halting is allowed, accesses to this register from EL1, EL2, and EL3 generate a trapped Software Access Debug event.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2.TDA](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TDA==1, accesses to this register from EL1 and EL2 are trapped to EL3.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

DBGWCR<n>_EL1, Debug Watchpoint Control Registers, n = 0 - 15

The DBGWCR<n>_EL1 characteristics are:

Purpose

Holds control information for a watchpoint. Forms watchpoint n together with value register [DBGWVR<n>_EL1](#).

This register is part of the Debug registers functional group.

Configuration

AArch64 System register DBGWCR<n>_EL1 is architecturally mapped to AArch32 System register [DBGWCR<n>](#).

AArch64 System register DBGWCR<n>_EL1 is architecturally mapped to External register [DBGWCR<n>_EL1](#).

If breakpoint n is not implemented then this register is unallocated.

This register is in the Cold reset domain. On a Cold reset RW fields in this register reset to architecturally UNKNOWN values. The register is not affected by a Warm reset.

Attributes

DBGWCR<n>_EL1 is a 32-bit register.

Field descriptions

The DBGWCR<n>_EL1 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0						0	0	0	WT																				

Bits [31:29]

Reserved, RES0.

MASK, bits [28:24]

Address mask. Only objects up to 2GB can be watched using a single mask.

MASK	Meaning
00000	No mask.
00001	Reserved.
00010	Reserved.

If programmed with a reserved value, a watchpoint must behave as if either:

- MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGWCRn_EL1.
- The watchpoint is disabled.

Software must not rely on this property because the behavior of reserved values might change in a future revision of the architecture.

Other values mask the corresponding number of address bits, from 0b00011 masking 3 address bits (0x00000007 mask for address) to 0b11111 masking 31 address bits (0x7FFFFFFF mask for address).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bits [23:21]

Reserved, RES0.

WT, bit [20]

Watchpoint type. Possible values are:

WT	Meaning
0	Unlinked data address match.
1	Linked data address match.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

LBN, bits [19:16]

Linked breakpoint number. For Linked data address watchpoints, this specifies the index of the Context-matching breakpoint linked to.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

SSC, bits [15:14]

Security state control. Determines the Security states under which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the HMC and PAC fields.

For more information on the operation of the SSC, HMC, and PAC fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

HMC, bit [13]

Higher mode control. Determines the debug perspective for deciding when a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and PAC fields.

For more information on the operation of the SSC, HMC, and PAC fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

BAS, bits [12:5]

Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by [DBGWVR<n>_EL1](#) is being watched.

BAS	Description
xxxxxxx1	Match byte at DBGWVR<n>_EL1
xxxxxx1x	Match byte at DBGWVR<n>_EL1+1
xxxxx1xx	Match byte at DBGWVR<n>_EL1+2
xxx1xxx	Match byte at DBGWVR<n>_EL1+3

In cases where [DBGWVR<n>_EL1](#) addresses a double-word:

BAS	Description, if DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at DBGWVR<n>_EL1+4
xx1xxxxx	Match byte at DBGWVR<n>_EL1+5
x1xxxxxx	Match byte at DBGWVR<n>_EL1+6
1xxxxxxx	Match byte at DBGWVR<n>_EL1+7

If [DBGWVR<n>_EL1\[2\]](#) == 1, only BAS[3:0] are used and BAS[7:4] are ignored. ARM deprecates setting [DBGWVR<n>_EL1\[2\]](#) == 1.

The valid values for BAS are non-zero binary numbers all of whose set bits are contiguous. All other values are reserved and must not be used by software. See 'Reserved DBGWCR<n>_EL1.BAS values' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

LSC, bits [4:3]

Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are:

LSC	Meaning
01	Match instructions that load from a watchpointed address.
10	Match instructions that store to a watchpointed address.
11	Match instructions that load from or store to a watchpointed address.

All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

PAC, bits [2:1]

Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated. This field must be interpreted along with the SSC and HMC fields.

For more information on the operation of the SSC, HMC, and PAC fields, see 'Execution conditions for which a watchpoint generates Watchpoint exceptions' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

E, bit [0]

Enable watchpoint n. Possible values are:

E	Meaning
0	Watchpoint disabled.
1	Watchpoint enabled.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Accessing the DBGWCR<n>_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
DBGWCR<n>_EL1	10	000	0000	n<3:0>	111

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
x	0	1	-	RW	RW	RW

x	1	1	-	n/a	RW	RW
---	---	---	---	-----	----	----

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [EDSCR.TDA](#)==1, and [OSLSR_EL1.OSLK](#)==0, and halting is allowed, accesses to this register from EL1, EL2, and EL3 generate a trapped Software Access Debug event state.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2.TDA](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3.TDA](#)==1, accesses to this register from EL1 and EL2 are trapped to EL3.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

DBGWVR<n>_EL1, Debug Watchpoint Value Registers, n = 0 - 15

The DBGWVR<n>_EL1 characteristics are:

Purpose

Holds a data address value for use in watchpoint matching. Forms watchpoint n together with control register [DBGWCR<n>_EL1](#).

This register is part of the Debug registers functional group.

Configuration

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

AArch64 System register DBGWVR<n>_EL1 is architecturally mapped to External register [DBGWVR<n>_EL1](#).

If breakpoint n is not implemented then this register is unallocated.

This register is in the Cold reset domain. On a Cold reset RW fields in this register reset to architecturally UNKNOWN values. The register is not affected by a Warm reset.

Attributes

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

Field descriptions

The DBGWVR<n>_EL1 bit assignments are:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32			
RESS[14:4]											VA[52:49]					VA[48:2]																		
VA[48:2]																															0		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			

RESS[14:4], bits [63:53]

Reserved, Sign extended. Hardware and software must treat this field as RES0 if the most significant bit of VA is 0 or RES0, and as RES1 if the most significant bit of VA is 1.

Hardware always ignores the value of these bits and it is IMPLEMENTATION DEFINED whether:

- The bits are hardwired to a copy of the most significant bit of VA, meaning writes to these bits are ignored, and reads to the bits always return the hardwired value.
- The value in those bits can be written, and reads will return the last value written. The value held in those bits is ignored by hardware.

VA[52:49], bits [52:49]

In ARMv8.3 and ARMv8.2:

Extension to VA[48:2]. See VA[48:2] for more details.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.1 and ARMv8.0:

Extension to RESS[14:4]. See RESS[14:4] for more details.

VA[48:2], bits [48:2]

Bits[48:2] of the address value for comparison.

When ARMv8.2-LVA is implemented, and 52-bit addresses and a 64KB translation granule are in use, VA[52:49] forms the upper part of the address value. Otherwise, VA[52:49] are RESS.

ARM deprecates setting [DBGWVR<n>_EL1\[2\]](#) == 1.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bits [1:0]

Reserved, RES0.

Accessing the DBGWVR<n>_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
DBGWVR<n>_EL1	10	000	0000	n<3:0>	110

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RW	n/a	RW
x	0	1	-	RW	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [EDSCR.TDA](#)==1, and [OSLSR_EL1.OSLK](#)==0, [and halting is allowed](#), accesses to this register from EL1, EL2, and EL3 [generate a trapped Software Access Debug debug event state](#).

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2.TDA](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3.TDA](#)==1, accesses to this register from EL1 and EL2 are trapped to EL3.

<u>SysReg_v83A_xml-00bet4</u> <u>(old)</u>	htmldiff from-	<u>(new)</u>
	SysReg_v83A_xml-00bet4	<u>SysReg_v83A_xml-00bet5</u>

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

HCR_EL2, Hypervisor Configuration Register

The HCR_EL2 characteristics are:

Purpose

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

This register is part of the Virtualization registers functional group.

Configuration

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

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

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

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

HCR_EL2 is a 64-bit register.

Field descriptions

The HCR_EL2 bit assignments are:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	AT	NV1	NV	API	APK	0	MIO	CNO
RW	TRVM	HCD	TDZ	TGET	TVM	TLB	TPU	PCP	TSW	TACR	TIDCPT	TSC	TID3	TID2	TID1	TID0	TWE	TW	DC	BSU	FB	VSEVI	VF			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	

Bits [63:45]

Reserved, RES0.

AT, bit [44]

In ARMv8.3:

Present only if ARMv8.3-NV is implemented.

Address Translation. Non-secure EL1 execution of following address translation instructions is trapped to EL2:

[AT S1E0R](#), [AT S1E0W](#), [AT S1E1R](#), [AT S1E1W](#), [AT S1E1RP](#), [AT S1E1WP](#)

AT	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 execution of the specified instructions is trapped to EL2.

If ARMv8.3-NV is not implemented, this field is RES0.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

NV1, bit [43]**In ARMv8.3:**

Present only if ARMv8.3-NV is implemented.

Nested virtualization. Non-secure EL1 accesses to registers [VBAR_EL1](#), [ELR_EL1](#), [SPSR_EL1](#) are trapped to EL2.

NV1	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 accesses to the specified registers are trapped to EL2.

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 the bits HCR_EL2.NV and HCR_EL2.NV1 are both set to 1 then following effects also apply:

- The Non-secure 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 programmed in that location.
- The Non-secure EL1 translation table Table descriptors, when Hierarchical Permissions are enabled: Bit[60] holds the PXNTable instead of the UXNTable, Bit[59] is RES0, Bit[61] is treated as 0 regardless of the actual value programmed in that location.
- When executing at Non-secure EL1 state, the PSTATE.PAN bit is treated as zero for all purposes except reading the value of the bit.
- The LDTR* and STTR* instructions are treated as the equivalent LDR* and STR* instructions, respectively.

This bit is permitted to be cached in a TLB.

If ARMv8.3-NV is not implemented, this field is RES0.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

NV, bit [42]**In ARMv8.3:**

Present only if ARMv8.3-NV is implemented.

Nested virtualization. Non-secure EL1 accesses to the special purpose or system registers or the execution of the EL1 or EL2 translation regime address translation and TLB maintenance instructions, are trapped to EL2.

NV	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 accesses to the specified registers or the execution of the specified instructions are trapped to EL2. Non-secure EL1 read accesses to the CurrentEL register return a value of 0x2.

The system or special purpose registers for which accesses are trapped are as follows:

- Registers accessed using MRS or MSR with a name ending in _EL2: [ACTLR_EL2](#), [AFSR0_EL2](#), [AFSR1_EL2](#), [AMAIR_EL2](#), [CNTHCTL_EL2](#), [CNTHP_CTL_EL2](#), [CNTHP_CVAL_EL2](#), [CNTHP_TVAL_EL2](#), [CNTHV_CTL_EL2](#), [CNTHV_CVAL_EL2](#), [CNTHV_TVAL_EL2](#), [CNTVOFF_EL2](#), [CONTEXTIDR_EL2](#), [CPTR_EL2](#), [DACR32_EL2](#), [DBGVCR32_EL2](#), [ELR_EL2](#), [ESR_EL2](#), [FAR_EL2](#), [FPEXC32_EL2](#), [HACR_EL2](#), [HCR_EL2](#), [HPFAR_EL2](#), [HSTR_EL2](#), [ICC_SRE_EL2](#), [ICH_AP0R<n>_EL2](#), [ICH_APIR<n>_EL2](#), [ICH_HCR_EL2](#), [ICH_LR<n>_EL2](#), [ICH_VMCRR_EL2](#), [IFSR32_EL2](#), [MAIR_EL2](#), [MDCR_EL2](#), [RMR_EL2](#), [SCTLR_EL2](#), [SPSR_EL2](#), [TCR_EL2](#), [TPIDR_EL2](#), [TTBR0_EL2](#), [TTBR1_EL2](#), [VBAR_EL2](#), [VMPIDR_EL2](#), [VPIDR_EL2](#), [VTCR_EL2](#), [VTTBR_EL2](#).
- Registers accessed using MRS or MSR with a name ending in _EL12: [AFSR0_EL1](#), [AFSR1_EL1](#), [AMAIR_EL1](#), [CNTKCTL_EL1](#), [CONTEXTIDR_EL1](#), [CPACR_EL1](#), [ELR_EL1](#), [ESR_EL1](#), [FAR_EL1](#), [MAIR_EL1](#), [SCTLR_EL1](#), [SPSR_EL1](#), [TCR_EL1](#), [TTBR0_EL1](#), [TTBR1_EL1](#), [VBAR_EL1](#).
- Registers accessed using MRS or MSR with a name ending in _EL02: [CNTP_CTL_EL0](#), [CNTP_CVAL_EL0](#), [CNTP_TVAL_EL0](#), [CNTV_CTL_EL0](#), [CNTV_CVAL_EL0](#), [CNTV_TVAL_EL0](#).

The priority of the trap to EL2 as a result of the above accesses is higher than any other resulting exception.

The following special purpose registers are trapped: SPSR_irq, SPSR_abt, SPSR_und, SPSR_fiq, [SP_EL1](#).

The instructions for which the execution is trapped are as follows:

- EL2 translation regime Address Translation instructions and TLB maintenance instructions: [AT S1E2R](#), [AT S1E2W](#), [TLBI ALLE2](#), [TLBI ALLE2IS](#), [TLBI VAE2](#), [TLBI VAE2IS](#), [TLBI VALE2](#), [TLBI VALE2IS](#).
- EL1 translation regime Address Translation instructions and TLB maintenance instructions that are only accessible from EL2 and above: [AT S12E0R](#), [AT S12E0W](#), [AT S12E1R](#), [AT S12E1W](#), [TLBI ALLE1](#), [TLBI ALLE1IS](#), [TLBI IPAS2E1](#), [TLBI IPAS2E1IS](#), [TLBI IPAS2LE1](#), [TLBI IPAS2LE1IS](#), [TLBI VMALLS12E1](#), [TLBI VMALLS12E1IS](#).
- ERET, ERETAA and ERETAB. The priority of this trap to EL2 is higher than the HCR_EL2.API bit.
- 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 bit is permitted to be cached in a TLB.

Note

Nested virtualization is supported for a Guest Hypervisor using any of below:

- Using HCR_EL2.E2H==1: Nested virtualization is simpler as it can have native access to its own memory management controls.
 - Using HCR_EL2.E2H==0: The Host Hypervisor should set HCR_EL2.TVM and CPTR_EL2.TCPAC to trap any Guest Hypervisor access to the Non-secure EL1 system registers which would be accesses for the Guest Guest OS.
-

If ARMv8.3-NV is not implemented, this field is RES0.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

API, bit [41]

In ARMv8.3:

Present only if ARMv8.3-TPAuth is implemented.

Controls the use of instructions related to Pointer Authentication:

- PACGA, XPACD, XPACI, and XPACLRI.
- AUTDA, AUTDB, AUTDZA, AUTDZB, AUTIA, AUTIA1716, AUTIASP, AUTIAZ, AUTIB, AUTIB1716, AUTIBSP, AUTIBZ, AUTIZA, AUTIZB, PACDA, PACDB, PACDZA, PACDZB, PACIA, PACIA1716, PACIASP, PACIAZ, PACIB, PACIB1716, PACIBSP, PACIBZ, PACIZA, PACIZ, RETAA, RETAB, BRAA, BRAB, BLRAA, BLRAB, BRAAZ, BRABZ, BLRAAZ, BLRABZ, ERETAA, ERETAB, LDRAA and LDRAB when enabled for the Non-secure EL1 translation regime (that is the associated SCTLR_EL1.En<N><M> ==1) in Non-secure EL0 when HCR_EL2.TGE==0 || HCR_EL2.E2H==0 or in Non-secure EL1.

Defined values are:

API	Meaning
0	Use of instructions related to Pointer Authentication in Non-secure EL0 when HCR_EL2.TGE==0 HCR_EL2.E2H==0, or in Non-secure EL1 when the instructions are enabled for the Non-secure EL1 translation regime, is trapped to EL2. 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.
1	This control does not cause any instructions to be trapped.

Note

If ARMv8.3-TPAuth is implemented but EL2 is not implemented, the system behaves as if this bit is 1.

If ARMv8.3-TPAuth is not implemented, this field is RES0.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

APK, bit [40]**In ARMv8.3:**

Present only if ARMv8.3-TPAuth is implemented.

Trap registers holding "key" values for Pointer Authentication. Traps accesses to the following registers from Non-secure EL1 to EL2:

- [APIAKeyLo_EL1](#), [APIAKeyHi_EL1](#), [APIBKeyLo_EL1](#), [APIBKeyHi_EL1](#), [APDAKeyLo_EL1](#), [APDAKeyHi_EL1](#), [APDBKeyLo_EL1](#), [APDBKeyHi_EL1](#), [APGAKeyLo_EL1](#), and [APGAKeyHi_EL1](#).

Defined values are:

APK	Meaning
0	Access to the registers holding "key" values for pointer authentication from non-secure EL1 are trapped to EL2.
1	This control does not cause any instructions to be trapped.

Note

If ARMv8.3-TPAuth is implemented but EL2 is not implemented, the system behaves as if this bit is 1.

If ARMv8.3-TPAuth is not implemented, this field is RES0.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

Bit [39]

Reserved, RES0.

MIOCNCE, bit [38]

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

MIOCNCE	Meaning
0	For the Non-secure EL1&0 translation regime, 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.
1	For the Non-secure EL1&0 translation regime, 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' in the ARMv8 ARM, section B2 (The AArch64 Application Level Memory Model).

This field can be implemented as RAZ/WI.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

When ARMv8.1-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.

TEA, bit [37]

Route synchronous External abort exceptions to EL2. If the RAS Extension is implemented, the possible values of this bit are:

TEA	Meaning
0	Does not route synchronous External abort exceptions from Non-secure EL0 and EL1 to EL2.
1	Route synchronous External abort exceptions from Non-secure EL0 and EL1 to EL2, if not routed to EL3.

This bit resets to zero on a Warm reset into AArch32 state.

When the RAS Extension is not implemented, this field is RES0.

TERR, bit [36]

Trap Error record accesses. If the RAS Extension is implemented, the possible values of this bit are:

TERR	Meaning
0	Does not trap accesses to error record registers from Non-secure EL1 to EL2.
1	Accesses to the ER* registers from Non-secure EL1 generate a Trap exception to EL2.

This bit resets to zero on a Warm reset into AArch32 state.

When the RAS Extension is not implemented, this field is RES0.

TLOR, bit [35]

In ARMv8.3, ARMv8.2 and ARMv8.1:

Trap LOR registers. Traps accesses to the [LORSA_EL1](#), [LOREA_EL1](#), [LORN_EL1](#), [LORC_EL1](#), and [LORID_EL1](#) registers from Non-secure EL1 to EL2.

TLOR	Meaning
0	This control does not cause any instructions to be trapped.
1	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.

In ARMv8.0:

Reserved, RES0.

E2H, bit [34]

In ARMv8.3, ARMv8.2 and ARMv8.1:

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
0	EL2 is running a hypervisor.
1	EL2 is running a Host Operating System.

For information on the behavior of this bit see Behavior of HCR_EL2.E2H.

This bit is permitted to be cached in a TLB.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

In ARMv8.0:

Reserved, RES0.

ID, bit [33]

Stage 2 Instruction access cacheability disable. For the Non-secure EL1&0 translation regime, when HCR_EL2.VM==1, this control forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable.

ID	Meaning
0	This control has no effect on stage 2 of the Non-secure EL1&0 translation regime.
1	For the Non-secure EL1&0 translation regime, forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

This bit has no effect on the EL2, EL2&0, or EL3 translation regimes.

When ARMv8.1-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.

CD, bit [32]

Stage 2 Data access cacheability disable. For the Non-secure EL1&0 translation regime, when 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
0	This control has no effect on stage 2 of the Non-secure EL1&0 translation regime for data accesses and translation table walks.
1	For the Non-secure EL1&0 translation regime, forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

This bit has no effect on the EL2, EL2&0, or EL3 translation regimes.

When ARMv8.1-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.

RW, bit [31]

Execution state control for lower Exception levels:

RW	Meaning
0	Lower levels are all AArch32.
1	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.

If all lower Exception levels cannot use AArch32 then this bit is RAO/WI.

In an implementation that includes EL3, when [SCR_EL3.NS](#)==0, 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 ARMv8.1-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.

TRVM, bit [30]

Trap Reads of Virtual Memory controls. Traps Non-secure EL1 reads of the virtual memory control registers to EL2, from both Execution states. The registers for which read accesses are trapped are as follows:

Non-secure EL1 using AArch64: [SCTLR_EL1](#), [TTBR0_EL1](#), [TTBR1_EL1](#), [TCR_EL1](#), [ESR_EL1](#), [FAR_EL1](#), [AFSR0_EL1](#), [AFSR1_EL1](#), [MAIR_EL1](#), [AMAIR_EL1](#), [CONTEXTIDR_EL1](#).

Non-secure EL1 using AArch32: [SCTLR](#), [TTBR0](#), [TTBR1](#), [TTBCR](#), [TTBCR2](#), [DACR](#), [DFSR](#), [IFSR](#), [DFAR](#), [IFAR](#), [ADFSR](#), [AIFSR](#), [PRRR](#), [NMRR](#), [MAIR0](#), [MAIR1](#), [AMAIR0](#), [AMAIR1](#), [CONTEXTIDR](#).

TRVM	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 read accesses to the specified Virtual Memory controls are trapped to EL2.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_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.

HCD, bit [29]

HVC instruction disable. Disables Non-secure state execution of HVC instructions, from both Execution states.

HCD	Meaning
0	HVC instruction execution is enabled at EL2 and Non-secure EL1.
1	HVC instructions are UNDEFINED at EL2 and Non-secure 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.

This bit is only implemented if EL3 is not implemented. Otherwise, it is RES0.

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.

TDZ, bit [28]

Trap [DC ZVA](#) instructions. Traps Non-secure EL0 and EL1 execution of [DC ZVA](#) instructions to EL2, from AArch64 state only.

TDZ	Meaning
0	This control does not cause any instructions to be trapped.
1	In AArch64 state, any attempt to execute a DC ZVA instruction at Non-secure EL1, or at Non-secure EL0 when the instruction is not UNDEFINED at EL0, is trapped to EL2. Reading the DCZID_EL0 returns a value that indicates that DC ZVA instructions are not supported.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

When ARMv8.1-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.

TGE, bit [27]

Trap General Exceptions, from Non-secure EL0.

TGE	Meaning
0	This control has no effect on execution at EL0.
1	<p>When the value of SCR_EL3.NS is 0, this control has no effect on execution at EL0.</p> <p>When the value of SCR_EL3.NS is 1, in all cases:</p> <ul style="list-style-type: none"> • All exceptions that would be routed to EL1 are routed to EL2. • The SCTLR_EL1.M field, or the SCTLR.M field if EL1 is using AArch32, is treated as being 0 for all purposes other than returning the result of a direct read of SCTLR_EL1 or 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. <p>When the value of SCR_EL3.NS is 1 and the value of HCR_EL2.E2H is 0, additionally:</p> <ul style="list-style-type: none"> • The HCR_EL2.{FMO, IMO, AMO} fields are treated as being 1 for all purposes other than a direct read or write access of HCR_EL2. • 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>For 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.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of [HCR_EL2](#).

TVM, bit [26]

Trap Virtual Memory controls. Traps Non-secure EL1 writes to the virtual memory control registers to EL2, from both Execution states. The registers for which write accesses are trapped are as follows:

Non-secure EL1 using AArch64: [SCTLR_EL1](#), [TTBR0_EL1](#), [TTBR1_EL1](#), [TCR_EL1](#), [ESR_EL1](#), [FAR_EL1](#), [AFSR0_EL1](#), [AFSR1_EL1](#), [MAIR_EL1](#), [AMAIR_EL1](#), [CONTEXTIDR_EL1](#).

Non-secure EL1 using AArch32: [SCTLR](#), [TTBR0](#), [TTBR1](#), [TTBCR](#), [TTBCR2](#), [DACR](#), [DFSR](#), [IFSR](#), [DFAR](#), [IFAR](#), [ADFSR](#), [AIFSR](#), [PRRR](#), [NMRR](#), [MAIR0](#), [MAIR1](#), [AMAIR0](#), [AMAIR1](#), [CONTEXTIDR](#).

TVM	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 write accesses to the specified EL1 virtual memory control registers are trapped to EL2.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of [HCR_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.

TTLB, bit [25]

Trap TLB maintenance instructions. Traps Non-secure EL1 execution of TLB maintenance instructions to EL2, from both Execution states. This applies to the following instructions:

- When Non-secure EL1 is using AArch64, [TLBI VMALLE1IS](#), [TLBI VAE1IS](#), [TLBI ASIDE1IS](#), [TLBI VAAE1IS](#), [TLBI VALE1IS](#), [TLBI VAALE1IS](#), [TLBI VMALLE1](#), [TLBI VAE1](#), [TLBI ASIDE1](#), [TLBI VAAE1](#), [TLBI VALE1](#), [TLBI VAALE1](#).
- When Non-secure EL1 is using AArch32, [TLBIALLIS](#), [TLBIMVAIS](#), [TLBIASIDIS](#), [TLBIMVAAIS](#), [TLBIMVALIS](#), [TLBIMVAALIS](#), [ITLBIALL](#), [ITLBIMVA](#), [ITLBIASID](#), [DTLBIALL](#), [DTLBIMVA](#), [DTLBIASID](#), [TLBIALL](#), [TLBIMVA](#), [TLBIASID](#), [TLBIMVAA](#), [TLBIMVAL](#), [TLBIMVAAL](#).

TTLB	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 execution of the specified TLB maintenance instructions are trapped to EL2.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of [HCR_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.

TPU, bit [24]

Trap cache maintenance instructions that operate to the Point of Unification. Traps execution of those cache maintenance instructions at Non-secure EL1 or EL0 using AArch64, and at Non-secure EL1 using AArch32, to EL2. This applies to the following instructions:

- When Non-secure EL0 is using AArch64, [IC IVAU](#), [DC CVAU](#). 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.
- When Non-secure EL1 is using AArch64, [IC IVAU](#), [IC IALLU](#), [IC IALLUIS](#), [DC CVAU](#).
- When Non-secure EL1 is using AArch32, [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
0	This control does not cause any instructions to be trapped.
1	Non-secure execution of the specified 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.

In an implementation that includes EL3, when the value of [SCR_EL3](#).NS is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

When ARMv8.1-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.

TPCP, bit [23]

In ARMv8.3 and ARMv8.2:

Trap data or unified cache maintenance instructions that operate to the Point of Coherency or Persistence. Traps execution of those cache maintenance instructions at Non-secure EL1 or EL0 using AArch64, and at Non-secure EL1 using AArch32, to EL2. This applies to the following instructions:

- When Non-secure EL0 is using AArch64, [DC CIVAC](#), [DC CVAC](#), [DC CVAP](#). 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.
- When Non-secure EL1 is using AArch64, [DC IVAC](#), [DC CIVAC](#), [DC CVAC](#), [DC CVAP](#).
- When Non-secure EL1 is using AArch32, [DCIMVAC](#), [DCCIMVAC](#), [DCCMVAC](#).

Note

- An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2. In addition:
 - [DC IVAC](#) is always UNDEFINED at EL0 using AArch64.
 - [DCIMVAC](#), [DCCIMVAC](#), and [DCCMVAC](#) are always UNDEFINED at EL0 using AArch32.
- In ARMv8.0 this field is named TPC. From ARMv8.2 it is named TPCP.

TPCP	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure execution of the specified 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.

In an implementation that includes EL3, when the value of [SCR_EL3](#).NS is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

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.

In ARMv8.1 and ARMv8.0:

Trap data or unified cache maintenance instructions that operate to the Point of Coherency. Traps execution of those cache maintenance instructions at Non-secure EL1 or EL0 using AArch64, and at Non-secure EL1 using AArch32, to EL2. This applies to the following instructions:

- When Non-secure EL0 is using AArch64, [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.
- When Non-secure EL1 is using AArch64, [DC IVAC](#), [DC CIVAC](#), [DC CVAC](#).
- When Non-secure EL1 is using AArch32, [DCIMVAC](#), [DCCIMVAC](#), [DCCMVAC](#).

Note

- An exception generated because an instruction is UNDEFINED at EL0 is higher priority than this trap to EL2. In addition:
 - [DC IVAC](#) is always UNDEFINED at EL0 using AArch64.
 - [DCIMVAC](#), [DCCIMVAC](#), and [DCCMVAC](#) are always UNDEFINED at EL0 using AArch32.
- In ARMv8.0 this field is named TPC. From ARMv8.2 it is named TPCP.

TPC	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure execution of the specified 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.

In an implementation that includes EL3, when the value of [SCR_EL3](#).NS is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

When ARMv8.1-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.

TSW, bit [22]

Trap data or unified cache maintenance instructions that operate by Set/Way. Traps execution of those cache maintenance instructions at Non-secure EL1 using AArch64, and at Non-secure EL1 using AArch32, to EL2. This applies to the following instructions:

- When Non-secure EL1 is using AArch64, [DC ISW](#), [DC CSW](#), [DC C1SW](#).
- When Non-secure EL1 is using AArch32, [DCISW](#), [DCCSW](#), [DCC1SW](#).

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
0	This control does not cause any instructions to be trapped.
1	Non-secure execution of the specified instructions is trapped to EL2.

In an implementation that includes EL3, when the value of [SCR_EL3](#).NS is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_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.

TACR, bit [21]

Trap Auxiliary Control Registers. Traps Non-secure EL1 accesses to the Auxiliary Control Registers to EL2, from both Execution states. This applies to the following register accesses:

- Non-secure EL1 using AArch64: [ACTLR_EL1](#).
- Non-secure EL1 using AArch32: [ACTLR](#) and, if implemented, [ACTLR2](#).

TACR	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 accesses to the specified registers are trapped to EL2.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_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.

TIDCP, bit [20]

Trap IMPLEMENTATION DEFINED functionality. Traps Non-secure EL1 accesses to the encodings reserved for IMPLEMENTATION DEFINED functionality to EL2. This applies to the following register accesses:

AArch64: The following reserved encoding spaces:

- 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.

AArch32: MCR and MRC instructions accessing the following encodings:

- 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 Non-secure EL0 is trapped to EL2. If it is not, then it is UNDEFINED, and any attempt to access it from Non-secure EL0 generates an exception that is taken to EL1.

TIDCP	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 accesses to or execution of the specified encodings reserved for IMPLEMENTATION DEFINED functionality are trapped to EL2.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

TSC, bit [19]

Trap SMC instructions. Traps Non-secure EL1 execution of SMC instructions to EL2, from both Execution states.

TSC	Meaning
0	This control does not cause any instructions to be trapped.
1	If EL3 is implemented, then any attempt to execute an SMC instruction at Non-secure EL1 using AArch64 or Non-secure EL1 using AArch32 is trapped to EL2, regardless of the value of SCR_EL3.SMD . If EL3 is not implemented, ARMv8.3-NV is implemented, and HCR_EL2.NV is 1, then any attempt to execute an SMC instruction at Non-secure EL1 using AArch64 is trapped to EL2.

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.

If EL3 is not implemented, and HCR_EL2.NV is 0, this bit is RES0.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_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.

TID3, bit [18]

Trap ID group 3. Traps Non-secure EL1 reads of the following registers to EL2:

AArch64: [ID_PFR0_EL1](#), [ID_PFR1_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_AA64MMFR2_EL1](#), [ID_AA64AFR0_EL1](#), [ID_AA64AFR1_EL1](#), [ID_AA64ZFR0_EL1](#) (where SVE is implemented), and [ID_MMFR4_EL1](#), except that if [ID_MMFR4_EL1](#) is implemented as RAZ/WI then it is IMPLEMENTATION DEFINED whether accesses to [ID_MMFR4_EL1](#) are trapped.

It is IMPLEMENTATION DEFINED whether this field traps MRS accesses to encodings in the following range that are not already mentioned in this field description:

- $Op0 = 3, Op1 = 0, CRn = c0, CRm = \{c2-c7\}, Op2 = \{0-7\}$.

AArch32: [ID_PFR0](#), [ID_PFR1](#), [ID_DFR0](#), [ID_AFR0](#), [ID_MMFR0](#), [ID_MMFR1](#), [ID_MMFR2](#), [ID_MMFR3](#), [ID_ISAR0](#), [ID_ISAR1](#), [ID_ISAR2](#), [ID_ISAR3](#), [ID_ISAR4](#), [ID_ISAR5](#), [MVFR0](#), [MVFR1](#), [MVFR2](#), and [ID_MMFR4](#), except that if [ID_MMFR4](#) is implemented as RAZ/WI then it is IMPLEMENTATION DEFINED whether accesses to [ID_MMFR4](#) are trapped.

MRC access to any of the following encodings are also trapped:

- $coproc=p15, opc1 = 0, CRn = c0, CRm = \{c3-c7\}, opc2 = \{0,1\}$.
- $coproc=p15, opc1 = 0, CRn = c0, CRm = c3, opc2 = 2$.
- $coproc=p15, opc1 = 0, CRn = c0, CRm = c5, opc2 = \{4,5\}$.

It is IMPLEMENTATION DEFINED whether this bit traps MRC accesses to the following encodings:

- $coproc=p15, opc1 = 0, CRn = c0, CRm = c2, opc2 = 7$.
- $coproc=p15, opc1 = 0, CRn = c0, CRm = c3, opc2 = \{3-7\}$.
- $coproc=p15, opc1 = 0, CRn = c0, CRm = \{c4, c6, c7\}, opc2 = \{2-7\}$.
- $coproc=p15, opc1 = 0, CRn = c0, CRm = c5, opc2 = \{2, 3, 6, 7\}$.

TID3	Meaning
0	This control does not cause any instructions to be trapped.
1	The specified Non-secure EL1 read accesses to ID group 3 registers are trapped to EL2.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_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.

TID2, bit [17]

Trap ID group 2. Traps the following register accesses to EL2:

AArch64:

- Non-secure EL1 reads of [CTR_EL0](#), [CCSIDR_EL1](#), [CCSIDR2_EL1](#), [CLIDR_EL1](#), and [CSSELR_EL1](#).
- Non-secure EL0 reads of [CTR_EL0](#), except that if the value of [SCTLR_EL1.UCT](#) is 0 then EL0 reads of [CTR_EL0](#) are UNDEFINED and any resulting exception takes precedence over this trap.
- Non-secure EL1 writes to [CSSELR_EL1](#).

AArch32:

- Non-secure EL1 reads of the [CTR](#), [CCSIDR](#), [CCSIDR2](#), [CLIDR](#), and [CSSELR](#).
- Non-secure EL1 writes to the [CSSELR](#).

TID2	Meaning
0	This control does not cause any instructions to be trapped.
1	The specified Non-secure EL1 and EL0 accesses to ID group 2 registers are trapped to EL2.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

When ARMv8.1-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.

TID1, bit [16]

Trap ID group 1. Traps Non-secure EL1 reads of the following registers are trapped to EL2:

AArch64: [REVIDR_EL1](#), [AIDR_EL1](#).

AArch32: [TCMTR](#), [TLBTR](#), [REVIDR](#), [AIDR](#).

TID1	Meaning
0	This control does not cause any instructions to be trapped.
1	The specified Non-secure EL1 read accesses to ID group 1 registers are trapped to EL2.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_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.

TID0, bit [15]

Trap ID group 0. Traps the following register accesses to EL2:

AArch64: None.

AArch32:

- Non-secure EL1 reads of the [JIDR](#).
- If the [JIDR](#) is RAZ from Non-secure EL0, Non-secure EL0 of the [JIDR](#).
- Non-secure EL1 reads of the [FPSID](#).

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
0	This control does not cause any instructions to be trapped.
1	The specified Non-secure EL1 read accesses to ID group 0 registers are trapped to EL2.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

In an AArch64-only implementation, this bit is RES0.

When ARMv8.1-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.

TWE, bit [14]

Traps Non-secure EL0 and EL1 execution of WFE instructions to EL2, from both Execution states.

TWE	Meaning
0	This control does not cause any instructions to be trapped.
1	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 or SCTLR_EL1.nTWE .

In AArch32 state, the attempted execution of a conditional WFE instruction is only trapped if the instruction passes its condition code check.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

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 ARMv8.1-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.

TWI, bit [13]

Traps Non-secure EL0 and EL1 execution of WFI instructions to EL2, from both Execution states.

TWI	Meaning
0	This control does not cause any instructions to be trapped.
1	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 or SCTLR_EL1.nTWI .

In AArch32 state, the attempted execution of a conditional WFI instruction is only trapped if the instruction passes its condition code check.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

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 ARMv8.1-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.

DC, bit [12]**Default Cacheability.**

DC	Meaning
0	This control has no effect on the Non-secure EL1&0 translation regime.
1	In Non-secure 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 field is permitted to be cached in a TLB.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

When ARMv8.1-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.

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
00	No effect
01	Inner Shareable
10	Outer Shareable
11	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.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

When ARMv8.1-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.

FB, bit [9]

Force broadcast. Causes the following instructions to be broadcast within the Inner Shareable domain when executed from Non-secure 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](#).

FB	Meaning
0	This field has no effect on the operation of the specified instructions.
1	When one of the specified instruction is executed at Non-secure EL1, the instruction is broadcast within the Inner Shareable shareability domain.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_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.

VSE, bit [8]

Virtual SError interrupt.

VSE	Meaning
0	This mechanism is not making a virtual SError interrupt pending.
1	A virtual SError interrupt is pending because of this mechanism.

The virtual SError interrupt is only enabled when the value of [HCR_EL2.{TGE, AMO}](#) is {0, 1}.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

VI, bit [7]

Virtual IRQ Interrupt.

VI	Meaning
0	This mechanism is not making a virtual IRQ pending.
1	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}.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

VF, bit [6]

Virtual FIQ Interrupt.

VF	Meaning
0	This mechanism is not making a virtual FIQ pending.
1	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}.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

AMO, bit [5]

Physical SError interrupt routing.

AMO	Meaning
0	When executing at Non-secure Exception levels below EL2, physical SError interrupts are not taken to EL2. When the value of HCR_EL2.TGE is 0, if the PE is executing at EL2 using AArch64, physical SError interrupts are not taken unless they are routed to EL3 by the SCR_EL3.EA bit. Virtual SError interrupts are disabled.
1	When executing at any Exception level in Non-secure state: <ul style="list-style-type: none"> Physical SError interrupts are taken to EL2 unless they are routed to EL3. If HCR_EL2.TGE==0 then virtual SError interrupts are enabled in the Non-secure state.

If the value of HCR_EL2.TGE is 1:

- Regardless of the value of the AMO bit, when executing in Non-secure state, physical asynchronous External aborts and SError interrupts target EL2 unless they are routed to EL3.
- When ARMv8.1-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 ARMv8.1-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' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

IMO, bit [4]

Physical IRQ Routing.

IMO	Meaning
0	When executing at Non-secure Exception levels below EL2, physical IRQ interrupts are not taken to EL2. When the value of HCR_EL2.TGE is 0, if the PE is executing at EL2 using AArch64, physical IRQ interrupts are not taken unless they are routed to EL3 by the SCR_EL3.IRQ bit. Virtual IRQ interrupts are disabled.
1	When executing at any Exception level in Non-secure state: <ul style="list-style-type: none"> Physical IRQ interrupts are taken to EL2 unless they are routed to EL3. If HCR_EL2.TGE==0 then Virtual IRQ interrupts are enabled in Non-secure state.

If the value of HCR_EL2.TGE is 1:

- Regardless of the value of the IMO bit, when executing in Non-secure state, physical IRQ Interrupts target EL2 unless they are routed to EL3.
- When ARMv8.1-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 ARMv8.1-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' in the ARMv8 ARM, section D1.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

FMO, bit [3]

Physical FIQ Routing.

FMO	Meaning
0	When executing at Non-secure Exception levels below EL2, physical FIQ interrupts are not taken to EL2. When the value of HCR_EL2.TGE is 0, if the PE is executing at EL2 using AArch64, physical FIQ interrupts are not taken unless they are routed to EL3 by the SCR_EL3.FIQ bit. Virtual FIQ interrupts are disabled.
1	When executing at any Exception level in Non-secure state: <ul style="list-style-type: none"> Physical FIQ interrupts are taken to EL2 unless they are routed to EL3. If HCR_EL2.TGE==0 then Virtual FIQ interrupts are enabled in Non-secure state.

If the value of HCR_EL2.TGE is 1:

- Regardless of the value of the FMO bit, when executing in Non-secure state, physical FIQ Interrupts target EL2 unless they are routed to EL3.
- When ARMv8.1-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 ARMv8.1-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' in the ARMv8 ARM, section D1.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

PTW, bit [2]

Protected Table Walk. In the Non-secure 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
0	The translation table walk occurs as if it is to Normal Non-cacheable memory. This means it can be made speculatively.
1	The memory access generates a stage 2 Permission fault.

This field is permitted to be cached in a TLB.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_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.

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
0	This control has no effect on the operation of data cache invalidate by set/way instructions.
1	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: [DCISW](#) performs the same invalidation as a [DCCISW](#) instruction.

This bit can be implemented as RES1.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_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.

VM, bit [0]

Virtualization enable. Enables stage 2 address translation for the Non-secure EL1&0 translation regime. Possible values of this bit are:

VM	Meaning
0	Non-secure EL1&0 stage 2 address translation disabled.
1	Non-secure EL1&0 stage 2 address translation enabled.

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_EL2.SWIO bit.

This bit is permitted to be cached in a TLB.

In an implementation that includes EL3, when the value of [SCR_EL3.NS](#) is 0 the PE behaves as if this field is 0 for all purposes other than a direct read or write access of HCR_EL2.

When ARMv8.1-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.

Accessing the HCR_EL2

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
HCR_EL2	11	100	0001	0001	000

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	RW
0	0	1	-	-	RW	RW
0	1	1	-	n/a	RW	RW
1	0	1	-	-	RW	RW
1	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and $\text{SCR_EL3.NS}=1$ && $\text{HCR_EL2.E2H}=0$:

- If [HCR_EL2.NV](#)=1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and $\text{SCR_EL3.NS}=1$ && $\text{HCR_EL2.E2H}=1$ && $\text{HCR_EL2.TGE}=0$:

- If [HCR_EL2.NV](#)=1, Non-secure accesses to this register from EL1 are trapped to EL2.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

<u>SysReg v83A xml-00bet4</u> <u>(old)</u>	htmldiff from-	<u>(new)</u>
	SysReg_v83A_xml-00bet4	<u>SysReg v83A xml-00bet5</u>

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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.

This register is part of:

- The Exception and fault handling registers functional group.
- The Virtualization registers functional group.

Configuration

AArch64 System register HPFAR_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HPFAR](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

HPFAR_EL2 is a 64-bit register.

Field descriptions

The HPFAR_EL2 bit assignments are:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FIPA[51:48]			FIPA[47:12]											
FIPA[47:12]																												0				0	0	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			

Execution at EL1 or EL0 makes HPFAR_EL2 become UNKNOWN.

Bits [63:44]

Reserved, RES0.

FIPA[51:48], bits [43:40]

In ARMv8.3 and ARMv8.2:

Extension to FIPA[47:12]. See FIPA[47:12] for more details.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

FIPA[47:12], bits [39:4]

Bits [47:12] of the faulting intermediate physical address. For implementations with fewer than 48 physical address bits, the corresponding upper bits in this field are RES0.

Bits [47:12] of the faulting intermediate physical address. When ARMv8.2-LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, FIPA[51:48] form the upper part of the address value. Otherwise, for implementations with fewer than 52 physical address bits, FIPA[51:48] are RES0.

When ARMv8.2-LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, FIPA[51:48] form the upper part of the address value. For implementations with fewer than 52 physical address bits, FIPA[51:48] are RES0.

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.

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

For all other exceptions taken to EL2, this register is UNKNOWN.

In an implementation or a translation granule that does not support ARMv8.2-LPA, the upper bits of this field are RES0.

Bits [3:0]

Reserved, RES0.

Accessing the HPFAR_EL2

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
HPFAR_EL2	11	100	0110	0000	100

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	RW
x	0	1	-	-	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.NV==1](#), Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.NV==1](#), Non-secure accesses to this register from EL1 are trapped to EL2.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

Configuration

There are no configuration notes.

Attributes

ID_AA64ISAR1_EL1 is a 64-bit register.

Field descriptions

The ID_AA64ISAR1_EL1 bit assignments are:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
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

Bits [63:32]

Reserved, RES0.

GPI, bits [31:28]

In ARMv8.3:

Indicates whether an IMPLEMENTATION DEFINED algorithm is implemented in the PE for generic code authentication, in AArch64 state. Defined values are:

GPI	Meaning
0000	Generic Authentication using an IMPLEMENTATION DEFINED algorithm is not implemented.
0001	Generic Authentication using an IMPLEMENTATION DEFINED algorithm is implemented. This involves the PACGA instruction.

All other values are reserved.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

GPA, bits [27:24]**In ARMv8.3:**

Indicates whether **QARMA** or **Prince** or Architected algorithm is implemented in the PE for generic code authentication, in AArch64 state. Defined values are:

GPA	Meaning
0000	Generic Authentication using an Architected algorithm is not implemented.
0001	Generic Authentication using the QARMA or Prince algorithm is implemented. This involves the PACGA instruction.

All other values are reserved.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

LRCPC, bits [23:20]**In ARMv8.3:**

Indicates support for weaker release consistency, RCpc based model. Defined values are:

LRCPC	Meaning
0000	The LDAPRB, LDAPRH and LDAPR instructions are not implemented.
0001	The LDAPRB, LDAPRH and LDAPR instructions are implemented.

All other values are reserved.

In ARMv8.0, ARMv8.1 and ARMv8.2 the only permitted value is 0000.

From ARMv8.3 the only permitted value is 0001. This feature is identified as ARMv8.3-RCPC.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

FCMA, bits [19:16]**In ARMv8.3:**

Indicates support for complex number addition and multiplication where numbers are stored in vectors. Defined values are:

FCMA	Meaning
0000	The FCMLA and FCADD instructions are not implemented.
0001	The FCMLA and FCADD instructions are implemented.

All other values are reserved.

In ARMv8.0, ARMv8.1 and ARMv8.2 the only permitted value is 0000.

From ARMv8.3 the only permitted value is 0001. This feature is identified as ARMv8.3-CompNum.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

JSCVT, bits [15:12]**In ARMv8.3:**

Indicates support for javascript conversion from double precision floating point values to integers in AArch64 state. Defined values are:

JSCVT	Meaning
0000	The FJCVTZS instruction is not implemented.
0001	The FJCVTZS instruction is implemented.

All other values are reserved.

In ARMv8.0, ARMv8.1 and ARMv8.2 the only permitted value is 0000.

From ARMv8.3 the only permitted value is 0001. This feature is identified as ARMv8.3-JSConv.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

API, bits [11:8]

In ARMv8.3:

Indicates whether an IMPLEMENTATION DEFINED algorithm is implemented in the PE for address authentication, in AArch64 state. Defined values are:

API	Meaning
0000	Address Authentication using an IMPLEMENTATION DEFINED algorithm is not implemented.
0001	Address Authentication using an IMPLEMENTATION DEFINED algorithm is implemented. This involves all Pointer Authentication instructions other than the PACGA instruction.

All other values are reserved.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

APA, bits [7:4]

In ARMv8.3:

Indicates whether **QARMA** or Architected algorithm is implemented in the PE for address authentication, in AArch64 state. Defined values are:

APA	Meaning
0000	Address Authentication using an Architected algorithm is not implemented.
0001	Address Authentication using the QARMA algorithm is implemented. This involves all Pointer Authentication instructions other than the PACGA instruction.

All other values are reserved.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

DPB, bits [3:0]

In ARMv8.3 and ARMv8.2:

Indicates support for the [DC CVAP](#) instruction in AArch64 state. Defined values are:

DPB	Meaning
0000	DC CVAP not supported.
0001	DC CVAP supported.

All other values are reserved.

ARMv8.2-DCPoP implements the functionality identified by the value 0001.

From ARMv8.2 the only permitted value is 0001.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

If API == 0000 and APA == 0000, then:

- 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 API == 0000 and APA == 0000 and GPI == 0000 and GPA == 0000, then:

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

Accessing the ID_AA64ISAR1_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64ISAR1_EL1	11	000	0000	0110	001

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RO	n/a	RO
x	0	1	-	RO	RO	RO
x	1	1	-	n/a	RO	RO

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2](#).TID3==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2](#).TID3==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

<u>SysReg_v83A_xml-00bet4</u> <u>(old)</u>	htmldiff from- SysReg_v83A_xml-00bet4	<u>(new)</u> <u>SysReg_v83A_xml-00bet5</u>
---	--	---

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

Configuration

The external register [EDPFR](#) gives information from this register.

Attributes

ID_AA64PFR0_EL1 is a 64-bit register.

Field descriptions

The ID_AA64PFR0_EL1 bit assignments are:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RAS				GIC				AdvSIMD				FP				EL3				EL2				EL1				SVE			
31	30	29	28	27	26	25	24	23	22	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.

SVE, bits [35:32]

In ARMv8.3 and ARMv8.2:

Scalable Vector Extension. Defined values are:

SVE	Meaning
0000	SVE is not implemented.
0001	SVE is implemented.

All other values are reserved.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

RAS, bits [31:28]

RAS Extension version. The defined values of this field are:

RAS	Meaning
0000	No RAS Extension.
0001	Version 1 of the RAS Extension present.

All other values are reserved.

GIC, bits [27:24]

System register GIC interface support. Defined values are:

GIC	Meaning
0000	No System register interface to the GIC is supported.
0001	System register interface to versions 3.0 and 4.0 of the GIC CPU interface is supported.

All other values are reserved.

AdvSIMD, bits [23:20]

Advanced SIMD. Defined values are:

AdvSIMD	Meaning
0000	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.
0001	As for 0000, and also includes support for half-precision floating-point arithmetic.
1111	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:

- 0000 in an implementation with Advanced SIMD support that does not include the ARMv8.2-FP16 extension.
- 0001 in an implementation with Advanced SIMD support that includes the ARMv8.2-FP16 extension.
- 1111 in an implementation without Advanced SIMD support.

FP, bits [19:16]

Floating-point. Defined values are:

FP	Meaning
0000	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.
0001	As for 0000, and also includes support for half-precision floating-point arithmetic.
1111	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:

- 0000 in an implementation with floating-point support that does not include the ARMv8.2-FP16 extension.
- 0001 in an implementation with floating-point support support that includes the ARMv8.2-FP16 extension.
- 1111 in an implementation without floating-point support.

EL3, bits [15:12]

EL3 Exception level handling. Defined values are:

EL3	Meaning
0000	EL3 is not implemented.
0001	EL3 can be executed in AArch64 state only.
0010	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
0000	EL2 is not implemented.
0001	EL2 can be executed in AArch64 state only.
0010	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
0001	EL1 can be executed in AArch64 state only.
0010	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
0001	EL0 can be executed in AArch64 state only.
0010	EL0 can be executed in either AArch64 or AArch32 state.

All other values are reserved.

Accessing the ID_AA64PFR0_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64PFR0_EL1	11	000	0000	0100	000

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RO	n/a	RO
x	0	1	-	RO	RO	RO

x	1	1	-	n/a	RO	RO
---	---	---	---	-----	----	----

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2](#).TID3==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2](#).TID3==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

28/09/2017 08:16:24

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg v83A xml-00bet4 (old)	htmldiff from-	(new)
	SysReg v83A xml-00bet4	SysReg v83A xml-00bet5

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

ID_MMFR3_EL1, AArch32 Memory Model Feature Register 3

The ID_MMFR3_EL1 characteristics are:

Purpose

Provides information about the implemented memory model and memory management support in AArch32 state.

Must be interpreted with [ID_MMFR0_EL1](#), [ID_MMFR1_EL1](#), [ID_MMFR2_EL1](#), and [ID_MMFR4_EL1](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

Configuration

AArch64 System register ID_MMFR3_EL1 is architecturally mapped to AArch32 System register [ID_MMFR3](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

Attributes

ID_MMFR3_EL1 is a 32-bit register.

Field descriptions

The ID_MMFR3_EL1 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Supersec				CMemSz				CohWalk				PAN				MaintBcst				BPMaint				CMaintSW				CMaintVA			

Supersec, bits [31:28]

Supersections. On a VMSA implementation, indicates whether Supersections are supported. Defined values are:

Supersec	Meaning
0000	Supersections supported.
1111	Supersections not supported.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 1111.

CMemSz, bits [27:24]

Cached Memory Size. Indicates the physical memory size supported by the caches. Defined values are:

CMemSz	Meaning
0000	4GB, corresponding to a 32-bit physical address range.
0001	64GB, corresponding to a 36-bit physical address range.
0010	1TB or more, corresponding to a 40-bit or larger physical address range.

All other values are reserved.

In ARMv8-A the permitted values are 0000, 0001, and 0010.

CohWalk, bits [23:20]

Coherent Walk. Indicates whether Translation table updates require a clean to the point of unification. Defined values are:

CohWalk	Meaning
0000	Updates to the translation tables require a clean to the point of unification to ensure visibility by subsequent translation table walks.
0001	Updates to the translation tables do not require a clean to the point of unification to ensure visibility by subsequent translation table walks.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

PAN, bits [19:16]

In ARMv8.3, ARMv8.2 and ARMv8.1:

Privileged Access Never. Indicates support for the PAN bit in [CPSR](#), [SPSR](#), and [DPSR](#) in AArch32 state. Defined values are:

PAN	Meaning
0000	PAN not supported.
0001	PAN supported.
0010	PAN supported and ATS1CPRP and ATS1CPWP instructions supported.

All other values are reserved.

ARMv8.1-PAN implements the functionality identified by the value 0001.

ARMv8.2-ATS1E1 implements the functionality added by the value 0010.

In ARMv8.1 the value ~~is~~ 0000 is not permitted.

From ARMv8.2, the only permitted value is 0010.

In ARMv8.0:

Reserved, RES0.

MaintBcst, bits [15:12]

Maintenance Broadcast. Indicates whether Cache, TLB, and branch predictor operations are broadcast. Defined values are:

MaintBcst	Meaning
0000	Cache, TLB, and branch predictor operations only affect local structures.
0001	Cache and branch predictor operations affect structures according to shareability and defined behavior of instructions. TLB operations only affect local structures.
0010	Cache, TLB, and branch predictor operations affect structures according to shareability and defined behavior of instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

BPMaint, bits [11:8]

Branch Predictor Maintenance. Indicates the supported branch predictor maintenance operations in an implementation with hierarchical cache maintenance operations. Defined values are:

BPMaint	Meaning
0000	None supported.
0001	Supported branch predictor maintenance operations are: <ul style="list-style-type: none"> Invalidate all branch predictors.
0010	As for 0001, and adds: <ul style="list-style-type: none"> Invalidate branch predictors by VA.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

CMaintSW, bits [7:4]

Cache Maintenance by Set/Way. Indicates the supported cache maintenance operations by set/way, in an implementation with hierarchical caches. Defined values are:

CMaintSW	Meaning
0000	None supported.
0001	Supported hierarchical cache maintenance instructions by set/way are: <ul style="list-style-type: none"> Invalidate data cache by set/way. Clean data cache by set/way. Clean and invalidate data cache by set/way.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

In a unified cache implementation, the data cache maintenance operations apply to the unified caches.

CMaintVA, bits [3:0]

Cache Maintenance by Virtual Address. Indicates the supported cache maintenance operations by VA, in an implementation with hierarchical caches. Defined values are:

CMaintVA	Meaning
0000	None supported.
0001	Supported hierarchical cache maintenance operations by VA are: <ul style="list-style-type: none"> Invalidate data cache by VA. Clean data cache by VA. Clean and invalidate data cache by VA. Invalidate instruction cache by VA. Invalidate all instruction cache entries.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

In a unified cache implementation, data cache maintenance operations apply to the unified caches, and the instruction cache maintenance instructions are not implemented.

Accessing the ID_MMFR3_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
ID_MMFR3_EL1	11	000	0000	0001	111

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RO	n/a	RO
x	0	1	-	RO	RO	RO
x	1	1	-	n/a	RO	RO

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.TID3](#)==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.TID3](#)==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

28/09/2017 08:16:24

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg_v83A_xml-00bet4 (old)	htmldiff from-	(new)
	SysReg_v83A_xml-00bet4	SysReg_v83A_xml-00bet5

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

MIDR_EL1, Main ID Register

The MIDR_EL1 characteristics are:

Purpose

Provides identification information for the PE, including an implementer code for the device and a device ID number.

This register is part of the Identification registers functional group.

Configuration

AArch64 System register MIDR_EL1 is architecturally mapped to AArch32 System register [MIDR](#).

AArch64 System register MIDR_EL1 is architecturally mapped to External register [MIDR_EL1](#).

Attributes

MIDR_EL1 is a 32-bit register.

Field descriptions

The MIDR_EL1 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Implementer								Variant				Architecture				PartNum								Revision							

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:

Hex representation	ASCII representation	Implementer
0x41	A	ARM Limited
0x42	B	Broadcom Corporation
0x43	C	Cavium Inc.
0x44	D	Digital Equipment Corporation
0x49	I	Infineon Technologies AG
0x4D	M	Motorola or Freescale Semiconductor Inc.
0x4E	N	NVIDIA Corporation
0x50	P	Applied Micro Circuits Corporation
0x51	Q	Qualcomm Inc.
0x56	V	Marvell International Ltd.
0x69	i	Intel Corporation

ARM can assign codes that are not published in this manual. All values not assigned by ARM are reserved and must not be used.

Variant, bits [23:20]

An IMPLEMENTATION DEFINED variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product.

Architecture, bits [19:16]

The permitted values of this field are:

Architecture	Meaning
0001	ARMv4
0010	ARMv4T
0011	ARMv5 (obsolete)
0100	ARMv5T
0101	ARMv5TE
0110	ARMv5TEJ
0111	ARMv6
1111	Architectural features are individually identified in the ID_* registers, see 'ID identification registers, functional group' in the ARMv8 ARM, section K12.3.3G4.18.1.

All other values are reserved.

PartNum, bits [15:4]

An IMPLEMENTATION DEFINED primary part number for the device.

On processors implemented by ARM, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently.

Revision, bits [3:0]

An IMPLEMENTATION DEFINED revision number for the device.

Accessing the MIDR_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
MIDR_EL1	11	000	0000	0000	000

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	RO	n/a	RO
x	0	1	-	RO	RO	RO
x	1	1	-	n/a	RO	RO

This table applies to all instructions that can access this register.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMCEID0_EL0, Performance Monitors Common Event Identification register 0

The PMCEID0_EL0 characteristics are:

Purpose

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0000 to 0x001F and 0x4000 to 0x401F.

When the value of a bit in the register is 1 the corresponding common event is implemented or counted.

Note

ARM recommends that, if a **common** event is never counted, the value of the corresponding register bit is 0.

For more information about the common events and the use of the PMCEIDn_EL0 registers see The section describing 'Event numbers and common events' in chapter D5 'The Performance Monitors Extension' of the ARM Architecture Reference Manual, for ARMv8-A architecture profile.

This register is part of the Performance Monitors registers functional group.

Configuration

AArch64 System register PMCEID0_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMCEID0](#).

AArch64 System register PMCEID0_EL0 bits [63:32] are architecturally mapped to AArch32 System register [PMCEID2](#).

AArch64 System register PMCEID0_EL0 bits [31:0] are architecturally mapped to External register [PMCEID0](#).

AArch64 System register PMCEID0_EL0 bits [63:32] are architecturally mapped to External register [PMCEID2](#).

Attributes

PMCEID0_EL0 is a 64-bit register.

Field descriptions

The PMCEID0_EL0 bit assignments are:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																IDhi[31:0]															
																ID[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

IDhi[31:0], bits [63:32]

In ARMv8.3, ARMv8.2 and ARMv8.1:

IDhi[n] corresponds to common event (0x4000 + n).

For each bit:

IDhi[31:0]	Meaning
0	The common event is not implemented, or not counted.
1	The common event is implemented.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n>_EL0 registers of that earlier version of the PMU architecture.

In ARMv8.0:

Reserved, RES0.

ID[31:0], bits [31:0]

ID[n] corresponds to common event n.

For each bit:

ID[31:0]	Meaning
0	The common event is not implemented, or not counted.
1	The common event is implemented.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n>_EL0 registers of that earlier version of the PMU architecture.

Accessing the PMCEID0_EL0

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
PMCEID0_EL0	11	011	1001	1100	110

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RO	RO	n/a	RO
x	0	1	RO	RO	RO	RO
x	1	1	RO	n/a	RO	RO

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR_EL0](#).EN==0, read accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2](#).TPM==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TPM==1, read accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/09/2017 08:16:24

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMCEID1_EL0, Performance Monitors Common Event Identification register 1

The PMCEID1_EL0 characteristics are:

Purpose

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0020 to 0x003F and 0x4020 to 0x403F.

When the value of a bit in the register is 1 the corresponding common event is implemented or counted.

Note

ARM recommends that, if a **common** event is never counted, the value of the corresponding register bit is 0.

For more information about the common events and the use of the PMCEIDn_EL0 registers see The section describing 'Event numbers and common events' in chapter D5 'The Performance Monitors Extension' of the ARM Architecture Reference Manual, for ARMv8-A architecture profile.

This register is part of the Performance Monitors registers functional group.

Configuration

AArch64 System register PMCEID1_EL0 bits [31:0] are architecturally mapped to AArch32 System register [PMCEID1](#).

AArch64 System register PMCEID1_EL0 bits [63:32] are architecturally mapped to AArch32 System register [PMCEID3](#).

AArch64 System register PMCEID1_EL0 bits [31:0] are architecturally mapped to External register [PMCEID1](#).

AArch64 System register PMCEID1_EL0 bits [63:32] are architecturally mapped to External register [PMCEID3](#).

Attributes

PMCEID1_EL0 is a 64-bit register.

Field descriptions

The PMCEID1_EL0 bit assignments are:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
																IDhi[63:32]															
																ID[63:32]															
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

IDhi[63:32], bits [63:32]

In ARMv8.3, ARMv8.2 and ARMv8.1:

IDhi[n] corresponds to common event (0x4000 + n).

For each bit:

IDhi[63:32]	Meaning
0	The common event is not implemented, or not counted.
1	The common event is implemented.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n>_EL0 registers of that earlier version of the PMU architecture.

In ARMv8.0:

Reserved, RES0.

ID[63:32], bits [31:0]

ID[n] corresponds to common event n.

For each bit:

ID[63:32]	Meaning
0	The common event is not implemented, or not counted.
1	The common event is implemented.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n>_EL0 registers of that earlier version of the PMU architecture.

Accessing the PMCEID1_EL0

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
PMCEID1_EL0	11	011	1001	1100	111

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RO	RO	n/a	RO
x	0	1	RO	RO	RO	RO
x	1	1	RO	n/a	RO	RO

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR_EL0](#).EN==0, read accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2](#).TPM==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TPM==1, read accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/09/2017 08:16:24

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMSELR_EL0, Performance Monitors Event Counter Selection Register

The PMSELR_EL0 characteristics are:

Purpose

Selects the current event counter [PMEVCNTR<n>](#) or the cycle counter, CCNT.

PMSELR_EL0 is used in conjunction with [PMXEVTYPER_EL0](#) to determine the event that increments a selected event counter, and the modes and states in which the selected counter increments.

It is also used in conjunction with [PMXVCNTR_EL0](#), to determine the value of a selected event counter.

This register is part of the Performance Monitors registers functional group.

Configuration

AArch64 System register PMSELR_EL0 is architecturally mapped to AArch32 System register [PMSELR](#).

This register is in the Warm reset domain. On a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

Attributes

PMSELR_EL0 is a 32-bit register.

Field descriptions

The PMSELR_EL0 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SEL				

Bits [31:5]

Reserved, RES0.

SEL, bits [4:0]

Selects event counter, [PMEVCNTR<n>](#), where n is the value held in this field. This value identifies which event counter is accessed when a subsequent access to [PMXEVTYPER_EL0](#) or [PMXVCNTR_EL0](#) occurs.

This field can take any value from 0 (0b00000) to (PMCR.N)-1, or 31 (0b11111).

When PMSELR_EL0.SEL is 0b11111, it selects the cycle counter and:

- A read of the [PMXEVTYPER_EL0](#) returns the value of [PMCCFILTR_EL0](#).
- A write of the [PMXEVTYPER_EL0](#) writes to [PMCCFILTR_EL0](#).
- A read or write of [PMXVCNTR_EL0](#) has CONSTRAINED UNPREDICTABLE effects, that can be one of the following:
 - Access to [PMXVCNTR_EL0](#) is UNDEFINED.
 - Access to [PMXVCNTR_EL0](#) behaves as a NOP.
 - Access to [PMXVCNTR_EL0](#) behaves as if the register is RAZ/WI.
 - Access to [PMXVCNTR_EL0](#) behaves as if the PMSELR_EL0.SEL field contains an UNKNOWN value.

If this field is set to a value greater than or equal to the number of implemented counters, but not equal to 31:

If this field is set to a value greater than or equal to the number of implemented counters, but not equal to 31, the results of access to [PMXEVTYPER_EL0](#) or [PMXEVNTR_EL0](#) are CONstrained UNPREDICTABLE, and can be one of the following:

- Access to [PMXEVTYPER_EL0](#) or [PMXEVNTR_EL0](#) behaves as if the register is RAZ/WI.
- Access to [PMXEVTYPER_EL0](#) or [PMXEVNTR_EL0](#) behaves as if the PMSELR_EL0.SEL field contains an UNKNOWN value.
- Access to [PMXEVTYPER_EL0](#) or [PMXEVNTR_EL0](#) behaves as if the PMSELR_EL0.SEL field contains 0b11111.
- Direct reads of this field return an UNKNOWN value.
- Access to [PMXEVTYPER_EL0](#) or [PMXEVNTR_EL0](#) is UNDEFINED.
- The results of access to [PMXEVTYPER_EL0](#) or [PMXEVNTR_EL0](#) are behaves as a NOP. CONstrained UNPREDICTABLE, and can be one of the following:
 - Access to [PMXEVTYPER_EL0](#) or [PMXEVNTR_EL0](#) is UNDEFINED.
 - Access to [PMXEVTYPER_EL0](#) or [PMXEVNTR_EL0](#) behaves as a NOP.
 - Access to [PMXEVTYPER_EL0](#) or [PMXEVNTR_EL0](#) behaves as if the register is RAZ/WI.
 - Access to [PMXEVTYPER_EL0](#) or [PMXEVNTR_EL0](#) behaves as if the PMSELR_EL0.SEL field contains an UNKNOWN value.
 - Access to [PMXEVTYPER_EL0](#) or [PMXEVNTR_EL0](#) behaves as if the PMSELR_EL0.SEL field contains 0b11111.

Direct reads of this field return an UNKNOWN value.

Accessing the PMSELR_EL0

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
PMSELR_EL0	11	011	1001	1100	101

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW
x	0	1	RW	RW	RW	RW
x	1	1	RW	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

In both Security states, and not dependent on other configuration bits:

- If [PMUSERENR_EL0](#).EN==0, and [PMUSERENR_EL0](#).ER==0, accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 :

- If [MDCR_EL2](#).TPM==1, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR_EL3](#).TPM==1, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
[\(old\)](#)

htmldiff from-
SysReg_v83A_xml-00bet4

[\(new\)](#)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

SCTLR_EL1, System Control Register (EL1)

The SCTLR_EL1 characteristics are:

Purpose

Provides top level control of the system, including its memory system, at EL1 and EL0.

This register is part of the Other system control registers functional group.

Configuration

AArch64 System register SCTLR_EL1 is architecturally mapped to AArch32 System register [SCTLR](#).

Some or all RW fields of this register have defined reset values. These apply only if the PE resets into EL1 using AArch64. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

Attributes

SCTLR_EL1 is a 32-bit register.

Field descriptions

The SCTLR_EL1 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	
EnIA	EnIBL	SMAO	EnTL	SMDO	EnDA	UCI	EE	E0E	SPAN	1	IESB	1	WXN	nTWE	0	nTWI	UCT	DZE	EnDB	I	1	0	UMA	SED	ITD	0	CP15BEN

EnIA, bit [31]

In ARMv8.3:

Controls enabling of pointer authentication (using the APIAKey_EL1 key) of instruction addresses in the EL1&0 translation regime.

Possible values of this bit are:

EnIA	Meaning
0	Pointer authentication (using the APIAKey_EL1 key) of instruction addresses is not enabled.
1	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.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

EnIB, bit [30]**In ARMv8.3:**

Controls enabling of pointer authentication (using the APIBKey_EL1 key) of instruction addresses in the EL1&0 translation regime.

Possible values of this bit are:

EnIB	Meaning
0	Pointer authentication (using the APIBKey_EL1 key) of instruction addresses is not enabled.
1	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.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

LSMAOE, bit [29]**In ARMv8.3 and ARMv8.2:**

Load Multiple and Store Multiple Atomicity and Ordering Enable. When the OPTIONAL feature ARMv8.2-LSMAOC is implemented, defined values are:

LSMAOE	Meaning
0	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.
1	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.

If this bit is not implemented, it is RES1.

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1,1}, this bit has no effect on execution at EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.1 and ARMv8.0:

Reserved, RES1.

nTLSMD, bit [28]**In ARMv8.3 and ARMv8.2:**

No Trap Load Multiple and Store Multiple to Device-nGRE/Device-nGnRE/Device-nGnRnE memory. When the OPTIONAL feature ARMv8.2-LSMAOC is implemented, defined values are:

nTLSMD	Meaning
0	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.
1	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.

If this bit is not implemented, it is RES1.

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1,1}, this bit has no effect on execution at EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.1 and ARMv8.0:

Reserved, RES1.

EnDA, bit [27]

In ARMv8.3:

Controls enabling of pointer authentication (using the APDAKey_EL1 key) of instruction addresses in the EL1&0 translation regime.

Possible values of this bit are:

EnDA	Meaning
0	Pointer authentication (using the APDAKey_EL1 key) of instruction addresses is not enabled.
1	Pointer authentication (using the APDAKey_EL1 key) of instruction 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.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

UCI, bit [26]

Traps EL0 execution of cache maintenance instructions to EL1, from AArch64 state only.

UCI	Meaning
0	Any attempt to execute a DC CVAU , DC CIVAC , DC CVAC , DC CVAP , or IC IVAU instruction at EL0 using AArch64 is trapped to EL1.
1	This control does not cause any instructions to be trapped.

When ARMv8.1-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.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

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
0	Explicit data accesses at EL1, and stage 1 translation table walks in the EL1&0 translation regime are little-endian.
1	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 ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on the PE.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to an IMPLEMENTATION DEFINED value.

E0E, bit [24]

Endianness of data accesses at EL0.

The possible values of this bit are:

E0E	Meaning
0	Explicit data accesses at EL0 are little-endian.
1	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 ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

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.

SPAN, bit [23]

In ARMv8.3, ARMv8.2 and ARMv8.1:

Set Privileged Access Never, on taking an exception to EL1.

SPAN	Meaning
0	PSTATE.PAN is set to 1 on taking an exception to EL1.
1	The value of PSTATE.PAN is left unchanged on taking an exception to EL1.

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.0:

Reserved, RES1.

Bit [22]

Reserved, RES1.

IESB, bit [21]

In ARMv8.3 and ARMv8.2:

Implicit error synchronization event enable. Possible values are:

IESB	Meaning
0	Disabled.
1	An implicit error synchronization event is added: <ul style="list-style-type: none"> • After 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 ~~operation~~ 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.

This field is part of ARMv8.2-IESB.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

Bit [20]

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
0	This control has no effect on memory access permissions.
1	Any region that is writable in the EL1&0 translation regime is forced to XN for accesses from software executing at EL1 or EL0.

The WXN bit is permitted to be cached in a TLB.

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on the PE.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

nTWE, bit [18]

Traps EL0 execution of WFE instructions to EL1, from both Execution states.

nTWE	Meaning
0	Any attempt to execute a WFE instruction at EL0 is trapped to EL1, if the instruction would otherwise have caused the PE to enter a low-power state.
1	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 ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bit [17]

Reserved, RES0.

nTWI, bit [16]

Traps EL0 execution of WFI instructions to EL1, from both Execution states.

nTWI	Meaning
0	Any attempt to execute a WFI instruction at EL0 is trapped EL1, if the instruction would otherwise have caused the PE to enter a low-power state.
1	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 of 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 ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

UCT, bit [15]

Traps EL0 accesses to the [CTR_EL0](#) to EL1, from AArch64 state only.

UCT	Meaning
0	Accesses to the CTR_EL0 from EL0 using AArch64 are trapped to EL1.
1	This control does not cause any instructions to be trapped.

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

DZE, bit [14]

Traps EL0 execution of [DC ZVA](#) instructions to EL1, from AArch64 state only.

DZE	Meaning
0	Any attempt to execute a DC ZVA instruction at EL0 using AArch64 is trapped to EL1. Reading DCZID_EL0 .DZP from EL0 returns 1, indicating that DC ZVA instructions are not supported.
1	This control does not cause any instructions to be trapped.

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

EnDB, bit [13]

In ARMv8.3:

Controls enabling of pointer authentication (using the APDBKey_EL1 key) of instruction addresses in the EL1&0 translation regime.

Possible values of this bit are:

EnDB	Meaning
0	Pointer authentication (using the APDBKey_EL1 key) of instruction addresses is not enabled.
1	Pointer authentication (using the APDBKey_EL1 key) of instruction 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.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

I, bit [12]

Instruction access Cacheability control, for accesses at EL0 and EL1:

I	Meaning
0	All instruction access to Normal memory from EL0 and EL1 are Non-cacheable for all levels of instruction and unified cache. 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.
1	This control has no effect on the Cacheability of instruction access to 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 ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on the PE.

When this register has an architecturally-defined reset value, this field resets to 0.

Bit [11]

Reserved, RES1.

Bit [10]

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, from AArch64 state only.

UMA	Meaning
0	Any attempt at EL0 using AArch64 to execute an MRS, MSR(register), or MSR(immediate) instruction that accesses the DAIF is trapped to EL1.
1	This control does not cause any instructions to be trapped.

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

SED, bit [8]

SETEND instruction disable. Disables SETEND instructions at EL0 using AArch32.

SED	Meaning
0	SETEND instruction execution is enabled at EL0 using AArch32.
1	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.

If EL0 cannot use AArch32, this bit is RES1.

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

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.

ITD, bit [7]

IT Disable. Disables some uses of IT instructions at EL0 using AArch32.

ITD	Meaning
0	All IT instruction functionality is enabled at EL0 using AArch32.
1	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 <code>hw1[3:0] != 1000</code>. All encodings of the subsequent instruction with the following values for <code>hw1</code>: <div style="margin-left: 20px;"> <p>11xxxxxxxxxxxxxx All 32-bit instructions, and the 16-bit instructions B, UDF, SVC, LDM, and STM.</p> <p>1011xxxxxxxxxxxxxx All instructions in 'Miscellaneous 16-bit instructions' in the ARMv8 ARM, section F3.2.5.</p> <p>10100xxxxxxxxxxxxx ADD Rd, PC, #imm</p> <p>01001xxxxxxxxxxxxx LDR Rd, [PC, #imm]</p> <p>0100x1xxx1111xxx ADD Rdn, PC; CMP Rn, PC; MOV Rd, PC; BX PC; BLX PC.</p> <p>010001xx1xxxx111 ADD PC, Rm; CMP PC, Rm; MOV PC, Rm. This pattern also covers UNPREDICTABLE cases with BLX Rn.</p> </div> <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. 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' in the ARMv8 ARM, section E1.2.4

If EL0 cannot use AArch32, this bit is RES1.

ITD is optional, but if it is implemented in the [SCTLR](#) then it must also be implemented in the SCTLR_EL1. If it is not implemented then this bit is RAZ/WI.

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

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.

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==1111) encoding space from EL0:

CP15BEN	Meaning
0	EL0 using AArch32: EL0 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is UNDEFINED.
1	EL0 using AArch32: EL0 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is enabled.

If EL0 cannot use AArch32, this bit is RES0.

CP15BEN is optional, but if it is implemented in the [SCTLR](#) then it must also be implemented in the SCTLR_EL1. If it is not implemented then this bit is RAO/WI.

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

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.

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 a SP alignment fault exception is generated. For more information, see 'SP alignment checking' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

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 a SP alignment fault exception is generated. For more information, see 'SP alignment checking' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on the PE.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

C, bit [2]

Cacheability control, for data accesses.

C	Meaning
0	All data access to Normal memory from EL0 and EL1, and all Normal memory accesses to the EL1&0 stage 1 translation tables, are Non-cacheable for all levels of data and unified cache.
1	This control has no effect on the 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 SCLTR.C. This means that Non-secure EL0 and Non-secure EL1 data accesses to Normal memory are Cacheable.

When ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on the PE.

When this register has an architecturally-defined reset value, 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
0	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.
1	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 ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on execution at EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

M, bit [0]

MMU enable for EL1 and EL0 stage 1 address translation. Possible values of this bit are:

M	Meaning
0	EL1 and EL0 stage 1 address translation disabled. See the SCTLR_EL1.I field for the behavior of instruction accesses to Normal memory.
1	EL1 and EL0 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 ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, this bit has no effect on the PE.

When this register has an architecturally-defined reset value, this field resets to 0.

Accessing the SCTLR_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
SCTLR_EL1	11	000	0001	0000	000
SCTLR_EL12	11	101	0001	0000	000

Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
SCTLR_EL1	x	x	0	-	RW	n/a	RW
SCTLR_EL1	0	0	1	-	RW	RW	RW
SCTLR_EL1	0	1	1	-	n/a	RW	RW
SCTLR_EL1	1	0	1	-	RW	SCTLR_EL2	RW
SCTLR_EL1	1	1	1	-	n/a	SCTLR_EL2	RW

SCTLR_EL12	x	x	0	-	-	n/a	-
SCTLR_EL12	0	0	1	-	-	-	-
SCTLR_EL12	0	1	1	-	n/a	-	-
SCTLR_EL12	1	0	1	-	-	RW	RW
SCTLR_EL12	1	1	1	-	n/a	RW	RW

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.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.TRVM](#)==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [HCR_EL2.TVM](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 using accessor SCTLR_EL12 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.TRVM](#)==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [HCR_EL2.TVM](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 using accessor SCTLR_EL12 are trapped to EL2.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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 ARMv8.1-VHE is implemented, and the value of [HCR_EL2](#).{E2H, TGE} is {1, 1}, these controls apply also to execution at Non-secure EL0.

This register is part of:

- The Virtualization registers functional group.
- The Other system control registers functional group.

Configuration

AArch64 System register SCTLR_EL2 is architecturally mapped to AArch32 System register [HSCTLR](#).

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

Some or all RW fields of this register have defined reset values. These apply only if the PE resets into EL2 using AArch64. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

Attributes

SCTLR_EL2 is a 32-bit register.

Field descriptions

The SCTLR_EL2 bit assignments are:

When HCR_EL2.{E2H, TGE} != {1, 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
EnIA	EnIB	1	1	EnDA	0	EE	0	1	1	IESB	0	WXN	1	0	1	0	0	EnDB	1	1	0	0	0	0	0	1	1	SA	C	A	M

This format applies in all ARMv8.0 implementations, and from ARMv8.1 in Secure state.

EnIA, bit [31]

In ARMv8.3:

Controls enabling of pointer authentication (using the APIAKey_EL1 key) of instruction addresses in the EL2&0 translation regime.

Possible values of this bit are:

EnIA	Meaning
0	Pointer authentication (using the APIAKey_EL1 key) of instruction addresses is not enabled.
1	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.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

EnIB, bit [30]

In ARMv8.3:

Controls enabling of pointer authentication (using the APIBKey_EL1 key) of instruction addresses in the EL2&0 translation regime.

Possible values of this bit are:

EnIB	Meaning
0	Pointer authentication (using the APIBKey_EL1 key) of instruction addresses is not enabled.
1	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.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

Bits [29:28]

Reserved, RES1.

EnDA, bit [27]

In ARMv8.3:

Controls enabling of pointer authentication (using the APDAKey_EL1 key) of instruction addresses in the EL2&0 translation regime.

Possible values of this bit are:

EnDA	Meaning
0	Pointer authentication (using the APDAKey_EL1 key) of instruction addresses is not enabled.
1	Pointer authentication (using the APDAKey_EL1 key) of instruction 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.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

Bit [26]

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.

The possible values of this bit are:

EE	Meaning
0	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.
1	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.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to an IMPLEMENTATION DEFINED value.

Bit [24]

Reserved, RES0.

Bits [23:22]

Reserved, RES1.

IESB, bit [21]

In ARMv8.3 and ARMv8.2:

Implicit error synchronization event enable. Possible values are:

IESB	Meaning
0	Disabled.
1	An implicit error synchronization event is added: <ul style="list-style-type: none"> After 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 ~~operation~~ 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.

This field is part of ARMv8.2-IESB.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

Bit [20]

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. The possible values of this bit are:

WXN	Meaning
0	This control has no effect on memory access permissions.
1	Any region that is writable in the EL2 or EL2&0 translation regime is forced to XN for accesses from software executing at EL2.

The WXN bit is permitted to be cached in a TLB.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bit [18]

Reserved, RES1.

Bit [17]

Reserved, RES0.

Bit [16]

Reserved, RES1.

Bits [15:14]

Reserved, RES0.

EnDB, bit [13]**In ARMv8.3:**

Controls enabling of pointer authentication (using the APDBKey_EL1 key) of instruction addresses in the EL2&0 translation regime.

Possible values of this bit are:

EnDB	Meaning
0	Pointer authentication (using the APDBKey_EL1 key) of instruction addresses is not enabled.
1	Pointer authentication (using the APDBKey_EL1 key) of instruction 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.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

I, bit [12]

Instruction access Cacheability control, for accesses at EL2:

I	Meaning
0	All instruction access to Normal memory from EL2 are Non-cacheable for all levels of instruction and unified cache. 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 Non-cacheable, Outer Non-cacheable memory.
1	This control has no effect on the Cacheability of instruction access to Normal memory from EL2. 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 EL1&0 or EL3 translation regimes.

When this register has an architecturally-defined reset value, this field resets to 0.

Bit [11]

Reserved, RES1.

Bits [10:6]

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 EL2 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' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

C, bit [2]

Cacheability control, for data accesses.

C	Meaning
0	All data access to Normal memory from EL2, and all Normal memory accesses to the EL2 translation tables, are Non-cacheable for all levels of data and unified cache.
1	This control has no effect on the Cacheability of: <ul style="list-style-type: none"> • Data access to Normal memory from EL2. • Normal memory accesses to the EL2 translation tables.

This bit has no effect on the EL1&0 or EL3 translation regimes.

When this register has an architecturally-defined reset value, this field resets to 0.

A, bit [1]

Alignment check enable. This is the enable bit for Alignment fault checking at EL2:

A	Meaning
0	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(s) being accessed.
1	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(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 this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

M, bit [0]

MMU enable for EL2 stage 1 address translation. Possible values of this bit are:

M	Meaning
0	EL2 stage 1 address translation disabled. See the SCTLR_EL2.I field for the behavior of instruction accesses to Normal memory.
1	EL2 stage 1 address translation enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

When HCR_EL2.{E2H, TGE} == {1, 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
EnIA	EnIB	LSMAOE	nT	LSMD	EnDA	UCI	EE	E0E	SPAN	1	IESB	1	WXN	nTWE	0	nTWI	UCTDZE	EnDB	I	1	0	0	SED	ITD	0	CP15BEN	SA0

This format applies only from ARMv8.1 and only in Non-secure state when [HCR_EL2](#).{E2H, TGE} == {1, 1}.

EnIA, bit [31]

In ARMv8.3:

Controls enabling of pointer authentication (using the APIAKey_EL1 key) of instruction addresses in the EL2&0 translation regime.

Possible values of this bit are:

EnIA	Meaning
0	Pointer authentication (using the APIAKey_EL1 key) of instruction addresses is not enabled.
1	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.

In ARMv8.2 and ARMv8.1:

Reserved, RES0.

EnIB, bit [30]

In ARMv8.3:

Controls enabling of pointer authentication (using the APIBKey_EL1 key) of instruction addresses in the EL2&0 translation regime.

Possible values of this bit are:

EnIB	Meaning
0	Pointer authentication (using the APIBKey_EL1 key) of instruction addresses is not enabled.
1	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.

In ARMv8.2 and ARMv8.1:

Reserved, RES0.

LSMAOE, bit [29]**In ARMv8.3 and ARMv8.2:**

Load Multiple and Store Multiple Atomicity and Ordering Enable. When the OPTIONAL feature ARMv8.2-LSMAOC is implemented, defined values are:

LSMAOE	Meaning
0	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.
1	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.

If this bit is not implemented, it is RES1.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.1:

Reserved, RES1.

nTLSMD, bit [28]**In ARMv8.3 and ARMv8.2:**

No Trap Load Multiple and Store Multiple to Device-nGRE/Device-nGnRE/Device-nGnRnE memory. When the OPTIONAL feature ARMv8.2-LSMAOC is implemented, defined values are:

nTLSMD	Meaning
0	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.
1	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.

If this bit is not implemented, it is RES1.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.1:

Reserved, RES1.

EnDA, bit [27]**In ARMv8.3:**

Controls enabling of pointer authentication (using the APDAKey_EL1 key) of instruction addresses in the EL2&0 translation regime.

Possible values of this bit are:

EnDA	Meaning
0	Pointer authentication (using the APDAKey_EL1 key) of instruction addresses is not enabled.
1	Pointer authentication (using the APDAKey_EL1 key) of instruction 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.

In ARMv8.2 and ARMv8.1:

Reserved, RES0.

UCI, bit [26]

Traps EL0 execution of cache maintenance instructions to EL2, from AArch64 state only.

UCI	Meaning
0	Any attempt to execute a DC CVAU , DC CIVAC , DC CVAC , or IC IVAU instruction at EL0 using AArch64 is trapped to EL2.
1	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.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

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 EL2&0 translation regime.

The possible values of this bit are:

EE	Meaning
0	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 EL2&0 translation regime are little-endian.
1	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 EL2&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 this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to an IMPLEMENTATION DEFINED value.

E0E, bit [24]

Endianness of data accesses at EL0.

The possible values of this bit are:

E0E	Meaning
0	Explicit data accesses at EL0 are little-endian.
1	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 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.

SPAN, bit [23]

Set Privileged Access Never, on taking an exception to EL2.

SPAN	Meaning
0	PSTATE.PAN is set to 1 on taking an exception to EL2.
1	The value of PSTATE.PAN is left unchanged on taking an exception to EL2.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bit [22]

Reserved, RES1.

IESB, bit [21]

In ARMv8.3 and ARMv8.2:

Implicit error synchronization event enable. Possible values are:

IESB	Meaning
0	Disabled.
1	An implicit error synchronization event is added: <ul style="list-style-type: none"> After 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 ~~operation~~ 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.

This field is part of ARMv8.2-IESB.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.1:

Reserved, RES0.

Bit [20]

Reserved, RES1.

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. The possible values of this bit are:

WXN	Meaning
0	This control has no effect on memory access permissions.
1	Any region that is writable in the EL2 or EL2&0 translation regime is forced to XN for accesses from software executing at EL2.

The WXN bit is permitted to be cached in a TLB.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

nTWE, bit [18]

Traps EL0 execution of WFE instructions to EL2, from both Execution states.

nTWE	Meaning
0	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.
1	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 of 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 this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bit [17]

Reserved, RES0.

nTWI, bit [16]

Traps EL0 execution of WFI instructions to EL2, from both Execution states.

nTWI	Meaning
0	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.
1	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 of 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 this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

UCT, bit [15]

Traps EL0 accesses to the [CTR_EL0](#) to EL2, from AArch64 state only.

UCT	Meaning
0	Accesses to the CTR_EL0 from EL0 using AArch64 are trapped to EL2.
1	This control does not cause any instructions to be trapped.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

DZE, bit [14]

Traps EL0 execution of [DC ZVA](#) instructions to EL2, from AArch64 state only.

DZE	Meaning
0	Any attempt to execute a DC ZVA instruction at EL0 using AArch64 is trapped to EL2. Reading DCZID_EL0.DZP from EL0 returns 1, indicating that DC ZVA instructions are not supported.
1	This control does not cause any instructions to be trapped.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

EnDB, bit [13]

In ARMv8.3:

Controls enabling of pointer authentication (using the APDBKey_EL1 key) of instruction addresses in the EL2&0 translation regime.

Possible values of this bit are:

EnDB	Meaning
0	Pointer authentication (using the APDBKey_EL1 key) of instruction addresses is not enabled.
1	Pointer authentication (using the APDBKey_EL1 key) of instruction 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.

In ARMv8.2 and ARMv8.1:

Reserved, RES0.

I, bit [12]

Instruction access Cacheability control, for accesses at EL2 and EL0:

I	Meaning
0	All instruction access to Normal memory from EL2 and EL0 are Non-cacheable for all levels of instruction and unified cache. If the value of SCTLR_EL2.M is 0, instruction accesses from stage 1 of the EL2&0 translation regime are to Normal, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable memory.
1	This control has no effect on the Cacheability of instruction access to Normal memory from EL2 and EL0. If the value of SCTLR_EL2.M is 0, instruction accesses from stage 1 of the 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 regimes.

When this register has an architecturally-defined reset value, this field resets to 0.

Bit [11]

Reserved, RES1.

Bits [10:9]

Reserved, RES0.

SED, bit [8]

SETEND instruction disable. Disables SETEND instructions at EL0 using AArch32.

SED	Meaning
0	SETEND instruction execution is enabled at EL0 using AArch32.
1	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.

If EL0 cannot use AArch32, this bit is RES1.

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.

ITD, bit [7]

IT Disable. Disables some uses of IT instructions at EL0 using AArch32.

ITD	Meaning
0	All IT instruction functionality is enabled at EL0 using AArch32.
1	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: <div> <div>11xxxxxxxxxxxx</div> <div>All 32-bit instructions, and the 16-bit instructions B, UDF, SVC, LDM, and STM.</div> </div> <div> <div>1011xxxxxxxxxxxx</div> <div>All instructions in 'Miscellaneous 16-bit instructions' in the ARMv8 ARM, section F3.2.5.</div> </div> <div> <div>10100xxxxxxxxxxx</div> <div>ADD Rd, PC, #imm</div> </div> <div> <div>01001xxxxxxxxxxx</div> <div>LDR Rd, [PC, #imm]</div> </div> <div> <div>0100x1xxx1111xxx</div> <div>ADD Rdn, PC; CMP Rn, PC; MOV Rd, PC; BX PC; BLX PC.</div> </div> <div> <div>010001xx1xxxx111</div> <div>ADD PC, Rm; CMP PC, Rm; MOV PC, Rm. This pattern also covers UNPREDICTABLE cases with BLX Rn.</div> </div> <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. 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' in the ARMv8 ARM, section E1.2.4

If EL0 cannot use AArch32, this bit is RES1.

ITD is optional, but if it is implemented in the [SCTLR](#) then it must also be implemented in the SCTLR_EL1. If it is not implemented then this bit is RAZ/WI.

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.

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==1111) encoding space from EL0:

CP15BEN	Meaning
0	EL0 using AArch32: EL0 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is UNDEFINED.
1	EL0 using AArch32: EL0 execution of the CP15DMB , CP15DSB , and CP15ISB instructions is enabled.

If EL0 cannot use AArch32, this bit is RES0.

CP15BEN is optional, but if it is implemented in the [SCTLR](#) then it must also be implemented in the SCTLR_EL1. If it is not implemented then this bit is RAO/WI.

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.

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 a SP alignment fault exception is generated. For more information, see 'SP alignment checking' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

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 a SP alignment fault exception is generated. For more information, see 'SP alignment checking' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

C, bit [2]

Cacheability control, for data accesses.

C	Meaning
0	All data access to Normal memory from EL2 and EL0, and all Normal memory accesses to the EL2&0 translation tables, are Non-cacheable for all levels of data and unified cache.
1	This control has no effect on the Cacheability of: <ul style="list-style-type: none"> Data access to Normal memory from EL2 and EL0. Normal memory accesses to the EL2&0 translation tables.

This bit has no effect on the EL3 translation regimes.

When this register has an architecturally-defined reset value, this field resets to 0.

A, bit [1]

Alignment check enable. This is the enable bit for Alignment fault checking at EL2 and EL0:

A	Meaning
0	Alignment fault checking disabled when executing at EL2 and 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.
1	Alignment fault checking enabled when executing at EL2 and 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 this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

M, bit [0]

MMU enable for EL2&0 stage 1 address translation. Possible values of this bit are:

M	Meaning
0	EL2&0 stage 1 address translation disabled. See the SCTLR_EL2.I field for the behavior of instruction accesses to Normal memory.
1	EL2&1 stage 1 address translation enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

Accessing the SCTLR_EL2

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
SCTLR_EL2	11	100	0001	0000	000
SCTLR_EL1	11	000	0001	0000	000

Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
SCTLR_EL2	x	x	0	-	-	n/a	RW
SCTLR_EL2	0	0	1	-	-	RW	RW
SCTLR_EL2	0	1	1	-	n/a	RW	RW
SCTLR_EL2	1	0	1	-	-	RW	RW
SCTLR_EL2	1	1	1	-	n/a	RW	RW
SCTLR_EL1	x	x	0	-	SCTLR_EL1	n/a	SCTLR_EL1
SCTLR_EL1	0	0	1	-	SCTLR_EL1	SCTLR_EL1	SCTLR_EL1
SCTLR_EL1	0	1	1	-	n/a	SCTLR_EL1	SCTLR_EL1
SCTLR_EL1	1	0	1	-	SCTLR_EL1	RW	SCTLR_EL1
SCTLR_EL1	1	1	1	-	n/a	RW	SCTLR_EL1

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.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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.

This register is part of the Other system control registers functional group.

Configuration

Some or all RW fields of this register have defined reset values. These apply only if the PE resets into EL3 using AArch64. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

Attributes

SCTLR_EL3 is a 32-bit register.

Field descriptions

The SCTLR_EL3 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EnIA	EnIB	1	1	EnDA	0	EE	0	1	1	IESB	0	WXN	1	0	1	0	0	EnDB	I	1	0	0	0	0	0	1	1	SA	C	A	M

EnIA, bit [31]

In ARMv8.3:

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
0	Pointer authentication (using the APIAKey_EL1 key) of instruction addresses is not enabled.
1	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.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

EnIB, bit [30]**In ARMv8.3:**

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
0	Pointer authentication (using the APIBKey_EL1 key) of instruction addresses is not enabled.
1	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.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

Bits [29:28]

Reserved, RES1.

EnDA, bit [27]**In ARMv8.3:**

Controls enabling of pointer authentication (using the APDAKey_EL1 key) of instruction addresses in the EL3 translation regime.

Possible values of this bit are:

EnDA	Meaning
0	Pointer authentication (using the APDAKey_EL1 key) of instruction addresses is not enabled.
1	Pointer authentication (using the APDAKey_EL1 key) of instruction 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.

In ARMv8.2, ARMv8.1 and ARMv8.0:

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.

The possible values of this bit are:

EE	Meaning
0	Explicit data accesses at EL3, and stage 1 translation table walks in the EL3 translation regime are little-endian.
1	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.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to an IMPLEMENTATION DEFINED value.

Bit [24]

Reserved, RES0.

Bits [23:22]

Reserved, RES1.

IESB, bit [21]

In ARMv8.3 and ARMv8.2:

Implicit error synchronization event enable. Possible values are:

IESB	Meaning
0	Disabled.
1	An implicit error synchronization event is added: <ul style="list-style-type: none"> After 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. If the Effective value of the field is 1, then an implicit error synchronization event **operation** 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.

This field is part of ARMv8.2-IESB.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

In ARMv8.1 and ARMv8.0:

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
0	This control has no effect on memory access permissions.
1	Any region that is writable in the EL3 translation regime is forced to XN for accesses from software executing at EL3.

The WXN bit is permitted to be cached in a TLB.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bit [18]

Reserved, RES1.

Bit [17]

Reserved, RES0.

Bit [16]

Reserved, RES1.

Bits [15:14]

Reserved, RES0.

EnDB, bit [13]

In ARMv8.3:

Controls enabling of pointer authentication (using the APDBKey_EL1 key) of instruction addresses in the EL3 translation regime.

Possible values of this bit are:

EnDB	Meaning
0	Pointer authentication (using the APDBKey_EL1 key) of instruction addresses is not enabled.
1	Pointer authentication (using the APDBKey_EL1 key) of instruction 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.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

I, bit [12]

Instruction access Cacheability control, for accesses at EL3:

I	Meaning
0	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.
1	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.

When this register has an architecturally-defined reset value, this field resets to 0.

Bit [11]

Reserved, RES1.

Bits [10:6]

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' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

C, bit [2]

Cacheability control, for data accesses.

C	Meaning
0	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.
1	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.

When this register has an architecturally-defined reset value, this field resets to 0.

A, bit [1]

Alignment check enable. This is the enable bit for Alignment fault checking at EL3:

A	Meaning
0	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.
1	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.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

M, bit [0]

MMU enable for EL3 stage 1 address translation. Possible values of this bit are:

M	Meaning
0	EL3 stage 1 address translation disabled. See the SCTLR_EL3.I field for the behavior of instruction accesses to Normal memory.
1	EL3 stage 1 address translation enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

Accessing the SCTLR_EL3

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
SCTLR_EL3	11	110	0001	0000	000

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	RW
x	0	1	-	-	-	RW
x	1	1	-	n/a	-	RW

This table applies to all instructions that can access this register.

28/09/2017 08:16:24

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

TCR_EL1, Translation Control Register (EL1)

The TCR_EL1 characteristics are:

Purpose

The control register for stage 1 of the EL1&0 translation regime.

This register is part of the Virtual memory control registers functional group.

Configuration

AArch64 System register TCR_EL1 bits [31:0] are architecturally mapped to AArch32 System register [TTBCR](#).

AArch64 System register TCR_EL1 bits [63:32] are architecturally mapped to AArch32 System register [TTBCR2](#).

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

TCR_EL1 is a 64-bit register.

Field descriptions

The TCR_EL1 bit assignments are:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43
0	0	0	0	0	0	0	0	0	NFD1	NFD0	TBID1	TBID0	HWU162	HWU161	HWU160	HWU159	HWU062	HWU061	HWU060	HWU059
TG1	SH1	ORGN1	IRGN1	EPD1	A1	T1SZ									TG0			SH0		ORG
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11

Any of the bits in TCR_EL1 are permitted to be cached in a TLB.

Bits [63:55]

Reserved, RES0.

NFD1, bit [54]

In ARMv8.3 and ARMv8.2:

Present only if SVE is implemented.

Non-fault translation table walk disable for translations using [TTBR1_EL1](#).

This bit controls whether to perform a translation table walk in response to an SVE non-fault access [from EL0](#) for an address that is translated using [TTBR1_EL1](#). The affected access types are:

- All accesses due to an SVE non-fault contiguous load instruction.
- Only the speculative accesses due to an SVE first-fault gather load. Speculative accesses due to an SVE first-fault contiguous load are not affected.
- Accesses due to prefetch instructions might be affected, but the effect is not architecturally visible.

See 'The Scalable Vector Extension (SVE)', in the ARM ARM, chapter A1 for more information.

Defined values are:

NFD1	Meaning
0	Perform translation table walks using TTBR1_EL1 .
1	A TLB miss on an address that is translated using TTBR1_EL1 due to an SVE non-fault access generates a Translation fault. No translation table walk is performed.

If SVE is not implemented, this field is RES0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

NFD0, bit [53]

In ARMv8.3 and ARMv8.2:

Present only if SVE is implemented.

Non-fault translation table walk disable for translations using [TTBR0_EL1](#).

This bit controls whether to perform a translation table walk in response to an SVE non-fault access [from EL0](#) for an address that is translated using [TTBR0_EL1](#). The affected access types are:

- All accesses due to an SVE non-fault contiguous load instruction.
- Only the speculative accesses due to an SVE first-fault gather load. Speculative accesses due to an SVE first-fault contiguous load are not affected.
- Accesses due to prefetch instructions might be affected, but the effect is not architecturally visible.

See 'The Scalable Vector Extension (SVE)', in the ARM ARM, chapter A1 for more information.

Defined values are:

NFD0	Meaning
0	Perform translation table walks using TTBR0_EL1 .
1	A TLB miss on an address that is translated using TTBR0_EL1 due to an SVE non-fault access generates a Translation fault. No translation table walk is performed.

If SVE is not implemented, this field is RES0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

TBID1, bit [52]

In ARMv8.3:

Present only if ARMv8.3-TPAuth is implemented.

Controls the use of the top byte of instruction addresses for address matching. Defined values are:

TBID1	Meaning
0	TCR_EL1.TBID1 applies to Instruction and Data accesses.
1	TCR_EL1.TBID1 applies to Data accesses only.

This affects addresses where the address would be translated by tables pointed to by [TTBR1_EL1](#).

If ARMv8.3-TPAuth is not implemented, this field is RES0.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

TBID0, bit [51]**In ARMv8.3:**

Present only if ARMv8.3-TPAuth is implemented.

Controls the use of the top byte of instruction addresses for address matching. Defined values are:

TBID0	Meaning
0	TCR_EL1.TBID0 applies to Instruction and Data accesses.
1	TCR_EL1.TBID0 applies to Data accesses only.

This affects addresses where the address would be translated by tables pointed to by [TTBR0_EL1](#).

If ARMv8.3-TPAuth is not implemented, this field is RES0.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

HWU162, bit [50]**In ARMv8.3 and ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL1](#).

Defined values are:

HWU162	Meaning
0	For translations using TTBR1_EL1 , bit[62] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR1_EL1 , bit[62] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD1 is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL1.HPD1 is 0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

HWU161, bit [49]**In ARMv8.3 and ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL1](#).

Defined values are:

HWU161	Meaning
0	For translations using TTBR1_EL1 , bit[61] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR1_EL1 , bit[61] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD1 is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL1.HPD1 is 0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

HWU160, bit [48]**In ARMv8.3 and ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL1](#).

Defined values are:

HWU160	Meaning
0	For translations using TTBR1_EL1 , bit[60] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR1_EL1 , bit[60] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD1 is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL1.HPD1 is 0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

HWU159, bit [47]**In ARMv8.3 and ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL1](#).

Defined values are:

HWU159	Meaning
0	For translations using TTBR1_EL1 , bit[59] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR1_EL1 , bit[59] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD1 is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL1.HPD1 is 0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

HWU062, bit [46]**In ARMv8.3 and ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

Defined values are:

HWU062	Meaning
0	For translations using TTBR0_EL1 , bit[62] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR0_EL1 , bit[62] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD0 is 1.

This bit is RES0 if ARMv8.2-TTBPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL1.HPD0 is 0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

HWU061, bit [45]

In ARMv8.3 and ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

Defined values are:

HWU061	Meaning
0	For translations using TTBR0_EL1 , bit[61] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR0_EL1 , bit[61] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD0 is 1.

This bit is RES0 if ARMv8.2-TTBPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL1.HPD0 is 0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

HWU060, bit [44]

In ARMv8.3 and ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

Defined values are:

HWU060	Meaning
0	For translations using TTBR0_EL1 , bit[60] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR0_EL1 , bit[60] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD0 is 1.

This bit is RES0 if ARMv8.2-TTBPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL1.HPD0 is 0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

HWU059, bit [43]**In ARMv8.3 and ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

Defined values are:

HWU059	Meaning
0	For translations using TTBR0_EL1 , bit[59] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR0_EL1 , bit[59] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL1.HPD0 is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL1.HPD0 is 0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

HPD1, bit [42]**In ARMv8.3, ARMv8.2 and ARMv8.1:**

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by [TTBR1_EL1](#).

Defined values are:

HPD1	Meaning
0	Hierarchical permissions are enabled.
1	Hierarchical permissions are disabled.

When disabled, the permissions are treated as if the bits are zero.

This bit is RES0 if ARMv8.1-HPD is not implemented.

In ARMv8.0:

Reserved, RES0.

HPD0, bit [41]**In ARMv8.3, ARMv8.2 and ARMv8.1:**

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by [TTBR0_EL1](#).

Defined values are:

HPD0	Meaning
0	Hierarchical permissions are enabled.
1	Hierarchical permissions are disabled.

When disabled, the permissions are treated as if the bits are zero.

This bit is RES0 if ARMv8.1-HPD is not implemented.

In ARMv8.0:

Reserved, RES0.

HD, bit [40]

In ARMv8.3, ARMv8.2 and ARMv8.1:

Hardware management of dirty state in stage 1 translations from EL0 and EL1.

Defined values are:

HD	Meaning
0	Stage 1 hardware management of dirty state disabled.
1	Stage 1 hardware management of dirty state enabled, only if the HA bit is also set to 1.

This bit is RES0 if ARMv8.1-TTHM is not implemented.

In ARMv8.0:

Reserved, RES0.

HA, bit [39]

In ARMv8.3, ARMv8.2 and ARMv8.1:

Hardware Access flag update in stage 1 translations from EL0 and EL1.

Defined values are:

HA	Meaning
0	Stage 1 Access flag update disabled.
1	Stage 1 Access flag update enabled.

This bit is RES0 if ARMv8.1-TTHM is not implemented.

In ARMv8.0:

Reserved, RES0.

TBI1, bit [38]

Top Byte ignored - indicates whether the top byte of an address is used for address match for the [TTBR1_EL1](#) region, or ignored and used for tagged addresses. Defined values are:

TBI1	Meaning
0	Top Byte used in the address calculation.
1	Top Byte ignored in the address calculation.

This affects addresses generated in EL0 and EL1 using AArch64 where the address would be translated by tables pointed to by [TTBR1_EL1](#). It has an effect whether the EL1&0 translation regime is enabled or not.

If ARMv8.3-TPAuth is implemented and TCR_EL1.TBID1 is 1, then this field only applies to Data accesses.

Otherwise, if the value of TBI1 is 1 and bit [55] of the target address to be stored to the PC is 1, then bits[63:56] of that target address are also set to 1 before the address is stored in the PC, in the following cases:

- A branch or procedure return within EL0 or EL1.
- An exception taken to EL1.
- An exception return to EL0 or EL1.

TBIO, bit [37]

Top Byte ignored - indicates whether the top byte of an address is used for address match for the [TTBR0_EL1](#) region, or ignored and used for tagged addresses. Defined values are:

TBIO	Meaning
0	Top Byte used in the address calculation.
1	Top Byte ignored in the address calculation.

This affects addresses generated in EL0 and EL1 using AArch64 where the address would be translated by tables pointed to by [TTBR0_EL1](#). It has an effect whether the EL1&0 translation regime is enabled or not.

If ARMv8.3-TPAuth is implemented and TCR_EL1.TBID0 is 1, then this field only applies to Data accesses.

Otherwise, if the value of TBIO is 1 and bit [55] of the target address to be stored to the PC is 1, then bits[63:56] of that target address are also set to 1 before the address is stored in the PC, in the following cases:

- A branch or procedure return within EL0 or EL1.
- An exception taken to EL1.
- An exception return to EL0 or EL1.

AS, bit [36]

ASID Size. Defined values are:

AS	Meaning
0	8 bit - the upper 8 bits of TTBR0_EL1 and TTBR1_EL1 are ignored by hardware for every purpose except reading back the register, and are treated as if they are all zeros for when used for allocation and matching entries in the TLB.
1	16 bit - the upper 16 bits of TTBR0_EL1 and TTBR1_EL1 are used for allocation and matching in the TLB.

If the implementation has only 8 bits of ASID, this field is RES0.

Bit [35]

Reserved, RES0.

IPS, bits [34:32]

Intermediate Physical Address Size.

IPS	Meaning
000	32 bits, 4GB.
001	36 bits, 64GB.
010	40 bits, 1TB.
011	42 bits, 4TB.
100	44 bits, 16TB.
101	48 bits, 256TB.
110	52 bits, 4PB

Other values are reserved.

The reserved values behave in the same way as the 101 encoding, but software must not rely on this property as the behavior of the reserved values might change in a future revision of the architecture.

The value 110 is permitted only if ARMv8.2-LPA is implemented and the translation granule size is 64KB.

In an implementation that supports 52-bit PAs, if the value of this field is not 110, then bits[51:48] of every translation table base address for the stage of translation controlled by TCR_EL1 are 0000.

TG1, bits [31:30]

Granule size for the [TTBR1_EL1](#).

TG1	Meaning
01	16KB
10	4KB
11	64KB

Other values are reserved.

If the value is programmed to either a reserved value, or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented for all purposes other than the value read back from this register.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

SH1, bits [29:28]

Shareability attribute for memory associated with translation table walks using [TTBR1_EL1](#). Defined values are:

SH1	Meaning
00	Non-shareable
10	Outer Shareable
11	Inner Shareable

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in AArch64 System registers and translation table entries' in the ARM ARM, section K1.2.2.

ORGN1, bits [27:26]

Outer cacheability attribute for memory associated with translation table walks using [TTBR1_EL1](#).

ORGN1	Meaning
00	Normal memory, Outer Non-cacheable
01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable
10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable
11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable

IRGN1, bits [25:24]

Inner cacheability attribute for memory associated with translation table walks using [TTBR1_EL1](#).

IRGN1	Meaning
00	Normal memory, Inner Non-cacheable
01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable
10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable
11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable

EPD1, bit [23]

Translation table walk disable for translations using [TTBR1_EL1](#). This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using [TTBR1_EL1](#). The encoding of this bit is:

EPD1	Meaning
0	Perform translation table walks using TTBR1_EL1 .
1	A TLB miss on an address that is translated using TTBR1_EL1 generates a Translation fault. No translation table walk is performed.

A1, bit [22]

Selects whether [TTBR0_EL1](#) or [TTBR1_EL1](#) defines the ASID. The encoding of this bit is:

A1	Meaning
0	TTBR0_EL1 .ASID defines the ASID.
1	TTBR1_EL1 .ASID defines the ASID.

T1SZ, bits [21:16]

The size offset of the memory region addressed by [TTBR1_EL1](#). The region size is $2^{(64-T1SZ)}$ bytes.

The maximum and minimum possible values for T1SZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

TG0, bits [15:14]

Granule size for the [TTBR0_EL1](#).

TG0	Meaning
00	4KB
01	64KB
10	16KB

Other values are reserved.

If the value is programmed to either a reserved value, or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented for all purposes other than the value read back from this register.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

SH0, bits [13:12]

Shareability attribute for memory associated with translation table walks using [TTBR0_EL1](#).

SH0	Meaning
00	Non-shareable
10	Outer Shareable
11	Inner Shareable

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in AArch64 System registers and translation table entries' in the ARM ARM, section K1.2.2.

ORGN0, bits [11:10]

Outer cacheability attribute for memory associated with translation table walks using [TTBR0_EL1](#).

ORGN0	Meaning
00	Normal memory, Outer Non-cacheable
01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable
10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable
11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable

IRGN0, bits [9:8]

Inner cacheability attribute for memory associated with translation table walks using [TTBR0_EL1](#).

IRGN0	Meaning
00	Normal memory, Inner Non-cacheable
01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable
10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable
11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable

EPD0, bit [7]

Translation table walk disable for translations using [TTBR0_EL1](#). This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using [TTBR0_EL1](#). The encoding of this bit is:

EPD0	Meaning
0	Perform translation table walks using TTBR0_EL1 .
1	A TLB miss on an address that is translated using TTBR0_EL1 generates a Translation fault. No translation table walk is performed.

Bit [6]

Reserved, RES0.

T0SZ, bits [5:0]

The size offset of the memory region addressed by [TTBR0_EL1](#). The region size is $2^{(64-T0SZ)}$ bytes.

The maximum and minimum possible values for T0SZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

Accessing the TCR_EL1

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
TCR_EL1	11	000	0010	0000	010
TCR_EL12	11	101	0010	0000	010

Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
TCR_EL1	x	x	0	-	RW	n/a	RW
TCR_EL1	0	0	1	-	RW	RW	RW
TCR_EL1	0	1	1	-	n/a	RW	RW
TCR_EL1	1	0	1	-	RW	TCR_EL2	RW
TCR_EL1	1	1	1	-	n/a	TCR_EL2	RW
TCR_EL12	x	x	0	-	-	n/a	-
TCR_EL12	0	0	1	-	-	-	-
TCR_EL12	0	1	1	-	n/a	-	-
TCR_EL12	1	0	1	-	-	RW	RW
TCR_EL12	1	1	1	-	n/a	RW	RW

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic TCR_EL1 or TCR_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2](#).TRVM==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [HCR_EL2](#).TVM==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR_EL2](#).NV==1, Non-secure accesses to this register from EL1 using accessor TCR_EL12 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2](#).TRVM==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [HCR_EL2](#).TVM==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR_EL2](#).NV==1, Non-secure accesses to this register from EL1 using accessor TCR_EL12 are trapped to EL2.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

TCR_EL2, Translation Control Register (EL2)

The TCR_EL2 characteristics are:

Purpose

The control register for stage 1 of the EL2, or EL2&0, translation regime:

- When the Effective value of [HCR_EL2.E2H](#) is 0, this register controls stage 1 of the EL2 translation regime, that supports a single VA range, translated using [TTBR0_EL2](#).
- When the value of [HCR_EL2.E2H](#) is 1, this register controls stage 1 of the EL2&0 translation regime, that supports both:
 - A lower VA range, translated using [TTBR0_EL2](#).
 - A higher VA range, translated using [TTBR1_EL2](#).

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

Configuration

AArch64 System register TCR_EL2 is architecturally mapped to AArch32 System register [HTCR](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

TCR_EL2 is a 64-bit register.

Field descriptions

The TCR_EL2 bit assignments are:

When HCR_EL2.E2H==0:

63	62	61	60		59		58		57		56		55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
0	0	0	0		0		0		0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	TBID	HWU62	HWU61	HWU60	HWU59	HPD	1	HD	HA	TBI	0	PS		TG0	SH0	ORGN0	IRGN0	0		0	T0SZ														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					

This format applies in Secure state, and in all ARMv8.0 implementations.

Any of the bits in TCR_EL2 are permitted to be cached in a TLB.

Bits [63:32]

Reserved, RES0.

Bit [31]

Reserved, RES1.

Bit [30]

Reserved, RES0.

TBID, bit [29]

In ARMv8.3:

Present only if ARMv8.3-TPAuth is implemented.

Controls the use of the top byte of instruction addresses for address matching. Defined values are:

TBID	Meaning
0	TCR_EL2.TBI applies to Instruction and Data accesses.
1	TCR_EL2.TBI applies to Data accesses only.

This affects addresses where the address would be translated by tables pointed to by [TTBR0_EL2](#).

If ARMv8.3-TPAuth is not implemented, this field is RES0.

In ARMv8.2, ARMv8.1 and ARMv8.0:

Reserved, RES0.

HWU62, bit [28]

In ARMv8.3 and ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table Block or Page entry.

Defined values are:

HWU62	Meaning
0	Bit[62] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	Bit[62] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL2.HPD is 0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

HWU61, bit [27]

In ARMv8.3 and ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table Block or Page entry.

Defined values are:

HWU61	Meaning
0	Bit[61] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	Bit[61] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD is 1.

This field is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL2.HPD is 0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

HWU60, bit [26]**In ARMv8.3 and ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table Block or Page entry.

Defined values are:

HWU60	Meaning
0	Bit[60] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	Bit[60] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL2.HPD is 0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

HWU59, bit [25]**In ARMv8.3 and ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table Block or Page entry.

Defined values are:

HWU59	Meaning
0	Bit[59] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	Bit[59] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL2.HPD is 0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

HPD, bit [24]**In ARMv8.3, ARMv8.2 and ARMv8.1:**

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by [TTBR0_EL2](#).

Defined values are:

HPD	Meaning
0	Hierarchical permissions are enabled.
1	Hierarchical permissions are disabled.

Note
In this case bit[61] (APTable[0]) and bit[59] (PXNTable) of the next level descriptor attributes are required to be ignored by the PE, and are no longer reserved, allowing them to be used by software.

When disabled, the permissions are treated as if the bits are zero.

This bit is RES0 if ARMv8.1-HPD is not implemented.

In ARMv8.0:

Reserved, RES0.

Bit [23]

Reserved, RES1.

HD, bit [22]

In ARMv8.3, ARMv8.2 and ARMv8.1:

Hardware management of dirty state in stage 1 translations from EL2.

Defined values are:

HD	Meaning
0	Stage 1 hardware management of dirty state disabled.
1	Stage 1 hardware management of dirty state enabled, only if the HA bit is also set to 1.

This bit is RES0 if ARMv8.1-TTHM is not implemented.

In ARMv8.0:

Reserved, RES0.

HA, bit [21]

In ARMv8.3, ARMv8.2 and ARMv8.1:

Hardware Access flag update in stage 1 translations from EL2.

Defined values are:

HA	Meaning
0	Stage 1 Access flag update disabled.
1	Stage 1 Access flag update enabled.

This bit is RES0 if ARMv8.1-TTHM is not implemented.

In ARMv8.0:

Reserved, RES0.

TBI, bit [20]

Top Byte ignored - indicates whether the top byte of an address is used for address match for the [TTBR0_EL2](#) region, or ignored and used for tagged addresses.

TBI	Meaning
0	Top Byte used in the address calculation.
1	Top Byte ignored in the address calculation.

This affects addresses generated in EL2 using AArch64 where the address would be translated by tables pointed to by [TTBR0_EL2](#). It has an effect whether the EL2, or EL2&0, translation regime is enabled or not.

If ARMv8.3-TPAuth is implemented and TCR_EL2.TBID is 1, then this field only applies to Data accesses.

Otherwise, if the value of TBI is 1, then bits[63:56] of that target address are also set to 0 before the address is stored in the PC, in the following cases:

- A branch or procedure return within EL2.
- An exception taken to EL2.
- An exception return to EL2.

Bit [19]

Reserved, RES0.

PS, bits [18:16]

Physical Address Size.

PS	Meaning
000	32 bits, 4GB.
001	36 bits, 64GB.
010	40 bits, 1TB.
011	42 bits, 4TB.
100	44 bits, 16TB.
101	48 bits, 256TB.
110	52 bits, 4PB

Other values are reserved.

The reserved values behave in the same way as the 101 encoding, but software must not rely on this property as the behavior of the reserved values might change in a future revision of the architecture.

The value 110 is permitted only if ARMv8.2-LPA is implemented and the translation granule size is 64KB.

In an implementation that supports 52-bit PAs, if the value of this field is not 110, then bits[51:48] of every translation table base address for the stage of translation controlled by TCR_EL2 are 0000.

TG0, bits [15:14]

Granule size for the [TTBR0_EL2](#).

TG0	Meaning
00	4KB
01	64KB
10	16KB

Other values are reserved.

If the value is programmed to either a reserved value, or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented for all purposes other than the value read back from this register.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

SH0, bits [13:12]

Shareability attribute for memory associated with translation table walks using [TTBR0_EL2](#).

SH0	Meaning
00	Non-shareable
10	Outer Shareable
11	Inner Shareable

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in AArch64 System registers and translation table entries' in the ARM ARM, section K1.2.2.

ORGN0, bits [11:10]

Outer cacheability attribute for memory associated with translation table walks using [TTBR0_EL2](#).

ORGN0	Meaning
00	Normal memory, Outer Non-cacheable
01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable
10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable
11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable

IRGN0, bits [9:8]

Inner cacheability attribute for memory associated with translation table walks using [TTBR0_EL2](#).

IRGN0	Meaning
00	Normal memory, Inner Non-cacheable
01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable
10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable
11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable

Bits [7:6]

Reserved, RES0.

T0SZ, bits [5:0]

The size offset of the memory region addressed by [TTBR0_EL2](#). The region size is $2^{(64-T0SZ)}$ bytes.

The maximum and minimum possible values for T0SZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

When HCR_EL2.E2H==1:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43
0	0	0	0	0	0	0	0	0	NFD1	NFD0	TBID1	TBID0	HWU162	HWU161	HWU160	HWU159	HWU062	HWU061	HWU060	HWU059
TG1	SH1	ORGN1	IRGN1	EPD1	A1	T1SZ										TG0		SH0		ORG
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11

This view of the register is only valid from ARMv8.1, in Non-secure state, when HCR_EL2.E2H is 1.

Any of the bits in TCR_EL2 are permitted to be cached in a TLB.

Bits [63:55]

Reserved, RES0.

NFD1, bit [54]**In ARMv8.3 and ARMv8.2:**

Present only if SVE is implemented.

Non-fault translation table walk disable for translations using [TTBR1_EL2](#).

This bit controls whether to perform a translation table walk in response to an SVE non-fault access [from EL0](#) for an address that is translated using [TTBR1_EL2](#). The affected access types are:

- All accesses due to an SVE non-fault contiguous load instruction.
- Only the speculative accesses due to an SVE first-fault gather load. Speculative accesses due to an SVE first-fault contiguous load are not affected.
- Accesses due to prefetch instructions might be affected, but the effect is not architecturally visible.

See 'The Scalable Vector Extension (SVE)', in the ARM ARM, chapter A1 for more information.

Defined values are:

NFD1	Meaning
0	Perform translation table walks using TTBR1_EL2 .
1	A TLB miss on an address that is translated using TTBR1_EL2 due to an SVE non-fault access generates a Translation fault. No translation table walk is performed.

If SVE is not implemented, this field is RES0.

In ARMv8.1:

Reserved, RES0.

NFD0, bit [53]**In ARMv8.3 and ARMv8.2:**

Present only if SVE is implemented.

Non-fault translation table walk disable for translations using [TTBR0_EL2](#).

This bit controls whether to perform a translation table walk in response to an SVE non-fault access [from EL0](#) for an address that is translated using [TTBR0_EL2](#). The affected access types are:

- All accesses due to an SVE non-fault contiguous load instruction.
- Only the speculative accesses due to an SVE first-fault gather load. Speculative accesses due to an SVE first-fault contiguous load are not affected.
- Accesses due to prefetch instructions might be affected, but the effect is not architecturally visible.

See 'The Scalable Vector Extension (SVE)', in the ARM ARM, chapter A1 for more information.

Defined values are:

NFD0	Meaning
0	Perform translation table walks using TTBR0_EL2 .
1	A TLB miss on an address that is translated using TTBR0_EL2 due to an SVE non-fault access generates a Translation fault. No translation table walk is performed.

If SVE is not implemented, this field is RES0.

In ARMv8.1:

Reserved, RES0.

TBID1, bit [52]**In ARMv8.3:**

Present only if ARMv8.3-TPAuth is implemented.

Controls the use of the top byte of instruction addresses for address matching. Defined values are:

TBID1	Meaning
0	TCR_EL2.TB11 applies to Instruction and Data accesses.
1	TCR_EL2.TB11 applies to Data accesses only.

This affects addresses where the address would be translated by tables pointed to by [TTBR1_EL2](#).

If ARMv8.3-TPAuth is not implemented, this field is RES0.

In ARMv8.2 and ARMv8.1:

Reserved, RES0.

TBID0, bit [51]

In ARMv8.3:

Present only if ARMv8.3-TPAuth is implemented.

Controls the use of the top byte of instruction addresses for address matching. Defined values are:

TBID0	Meaning
0	TCR_EL2.TB10 applies to Instruction and Data accesses.
1	TCR_EL2.TB10 applies to Data accesses only.

This affects addresses where the address would be translated by tables pointed to by [TTBR0_EL2](#).

If ARMv8.3-TPAuth is not implemented, this field is RES0.

In ARMv8.2 and ARMv8.1:

Reserved, RES0.

HWU162, bit [50]

In ARMv8.3 and ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL2](#).

Defined values are:

HWU162	Meaning
0	For translations using TTBR1_EL2 , bit[62] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR1_EL2 , bit[62] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD1 is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL2.HPD1 is 0.

In ARMv8.1:

Reserved, RES0.

HWU161, bit [49]**In ARMv8.3 and ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL2](#).

Defined values are:

HWU161	Meaning
0	For translations using TTBR1_EL2 , bit[61] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR1_EL2 , bit[61] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD1 is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL2.HPD1 is 0.

In ARMv8.1:

Reserved, RES0.

HWU160, bit [48]**In ARMv8.3 and ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL2](#).

Defined values are:

HWU160	Meaning
0	For translations using TTBR1_EL2 , bit[60] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR1_EL2 , bit[60] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD1 is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL2.HPD1 is 0.

In ARMv8.1:

Reserved, RES0.

HWU159, bit [47]**In ARMv8.3 and ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table Block or Page entry for translations using [TTBR1_EL2](#).

Defined values are:

HWU159	Meaning
0	For translations using TTBR1_EL2 , bit[59] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR1_EL2 , bit[59] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD1 is 1.

This bit is RES0 if ARMv8.2-TTBPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL2.HPD1 is 0.

In ARMv8.1:

Reserved, RES0.

HWU062, bit [46]

In ARMv8.3 and ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

Defined values are:

HWU062	Meaning
0	For translations using TTBR0_EL1 , bit[62] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR0_EL1 , bit[62] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD0 is 1.

This bit is RES0 if ARMv8.2-TTBPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL2.HPD0 is 0.

In ARMv8.1:

Reserved, RES0.

HWU061, bit [45]

In ARMv8.3 and ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

Defined values are:

HWU061	Meaning
0	For translations using TTBR0_EL1 , bit[61] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR0_EL1 , bit[61] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD0 is 1.

This bit is RES0 if ARMv8.2-TTBPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL2.HPD0 is 0.

In ARMv8.1:

Reserved, RES0.

HWU060, bit [44]

In ARMv8.3 and ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

Defined values are:

HWU060	Meaning
0	For translations using TTBR0_EL1 , bit[60] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR0_EL1 , bit[60] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD0 is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL2.HPD0 is 0.

In ARMv8.1:

Reserved, RES0.

HWU059, bit [43]

In ARMv8.3 and ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table Block or Page entry for translations using [TTBR0_EL1](#).

Defined values are:

HWU059	Meaning
0	For translations using TTBR0_EL1 , bit[59] of each stage 1 translation table Block or Page entry cannot be used by hardware for an IMPLEMENTATION DEFINED purpose.
1	For translations using TTBR0_EL1 , bit[59] of each stage 1 translation table Block or Page entry can be used by hardware for an IMPLEMENTATION DEFINED purpose if the value of TCR_EL2.HPD0 is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

The Effective value of this field is 0 if the value of TCR_EL2.HPD0 is 0.

In ARMv8.1:

Reserved, RES0.

HPD1, bit [42]

Hierarchical Permission Disables. This affects the hierarchical control bits, APTTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by [TTBR1_EL2](#).

Defined values are:

HPD1	Meaning
0	Hierarchical permissions are enabled.
1	Hierarchical permissions are disabled.

When disabled, the permissions are treated as if the bits are zero.

This bit is RES0 if ARMv8.1-HPD is not implemented.

HPD0, bit [41]

Hierarchical Permission Disables. This affects the hierarchical control bits, APTTable, PXNTable, and UXNTable, except NSTable, in the translation tables pointed to by [TTBR0_EL2](#).

Defined values are:

HPD0	Meaning
0	Hierarchical permissions are enabled.
1	Hierarchical permissions are disabled.

When disabled, the permissions are treated as if the bits are zero.

This bit is RES0 if ARMv8.1-HPD is not implemented.

HD, bit [40]

Hardware management of dirty state in stage 1 translations from EL2.

Defined values are:

HD	Meaning
0	Stage 1 hardware management of dirty state disabled.
1	Stage 1 hardware management of dirty state enabled, only if the HA bit is also set to 1.

This bit is RES0 if ARMv8.1-TTHM is not implemented.

HA, bit [39]

Hardware Access flag update in stage 1 translations from EL2.

Defined values are:

HA	Meaning
0	Stage 1 Access flag update disabled.
1	Stage 1 Access flag update enabled.

This bit is RES0 if ARMv8.1-TTHM is not implemented.

TBI1, bit [38]

Top Byte ignored - indicates whether the top byte of an address is used for address match for the [TTBR1_EL2](#) region, or ignored and used for tagged addresses. Defined values are:

TBI1	Meaning
0	Top Byte used in the address calculation.
1	Top Byte ignored in the address calculation.

This affects addresses generated in EL0 and EL2 using AArch64 where the address would be translated by tables pointed to by [TTBR1_EL2](#). It has an effect whether the EL2, or EL2&0, translation regime is enabled or not.

If ARMv8.3-TPAuth is implemented and TCR_EL2.TBID1 is 1, then this field only applies to Data accesses.

Otherwise, if the value of TBI1 is 1 and bit [55] of the target address to be stored to the PC is 1, then bits[63:56] of that target address are also set to 1 before the address is stored in the PC, in the following cases:

- A branch or procedure return within EL0 or EL1.
- An exception taken to EL1.
- An exception return to EL0 or EL1.

TBI0, bit [37]

Top Byte ignored - indicates whether the top byte of an address is used for address match for the [TTBR0_EL2](#) region, or ignored and used for tagged addresses. Defined values are:

TBI0	Meaning
0	Top Byte used in the address calculation.
1	Top Byte ignored in the address calculation.

This affects addresses generated in EL0 and EL2 using AArch64 where the address would be translated by tables pointed to by [TTBR0_EL2](#). It has an effect whether the EL2, or EL2&0, translation regime is enabled or not.

If ARMv8.3-TPAuth is implemented and TCR_EL2.TBID0 is 1, then this field only applies to Data accesses.

Otherwise, if the value of TBID0 is 1 and bit [55] of the target address to be stored to the PC is 1, then bits[63:56] of that target address are also set to 1 before the address is stored in the PC, in the following cases:

- A branch or procedure return within EL0 or EL1.
- An exception taken to EL1.
- An exception return to EL0 or EL1.

AS, bit [36]

ASID Size. Defined values are:

AS	Meaning
0	8 bit - the upper 8 bits of TTBR0_EL2 and TTBR1_EL2 are ignored by hardware for every purpose except reading back the register, and are treated as if they are all zeros for when used for allocation and matching entries in the TLB.
1	16 bit - the upper 16 bits of TTBR0_EL2 and TTBR1_EL2 are used for allocation and matching in the TLB.

If the implementation has only 8 bits of ASID, this field is RES0.

Bit [35]

Reserved, RES0.

IPS, bits [34:32]

Intermediate Physical Address Size.

IPS	Meaning
000	32 bits, 4GB.
001	36 bits, 64GB.
010	40 bits, 1TB.
011	42 bits, 4TB.
100	44 bits, 16TB.
101	48 bits, 256TB.
110	52 bits, 4PB.

Other values are reserved.

The reserved values behave in the same way as the 101 encoding, but software must not rely on this property as the behavior of the reserved values might change in a future revision of the architecture.

The value 110 is permitted only if ARMv8.2-LPA is implemented and the translation granule size is 64KB.

In an implementation that supports 52-bit PAs, if the value of this field is not 110, then bits[51:48] of every translation table base address for the stage of translation controlled by TCR_EL2 are 0000.

TG1, bits [31:30]

Granule size for the [TTBR1_EL2](#).

TG1	Meaning
01	16KB
10	4KB
11	64KB

Other values are reserved.

If the value is programmed to either a reserved value, or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented for all purposes other than the value read back from this register.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

SH1, bits [29:28]

Shareability attribute for memory associated with translation table walks using [TTBR1_EL2](#). Defined values are:

SH1	Meaning
00	Non-shareable
10	Outer Shareable
11	Inner Shareable

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in AArch64 System registers and translation table entries' in the ARM ARM, section K1.2.2.

ORGN1, bits [27:26]

Outer cacheability attribute for memory associated with translation table walks using [TTBR1_EL2](#).

ORGN1	Meaning
00	Normal memory, Outer Non-cacheable
01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable
10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable
11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable

IRGN1, bits [25:24]

Inner cacheability attribute for memory associated with translation table walks using [TTBR1_EL2](#).

IRGN1	Meaning
00	Normal memory, Inner Non-cacheable
01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable
10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable
11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable

EPD1, bit [23]

Translation table walk disable for translations using [TTBR1_EL2](#). This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using [TTBR1_EL2](#). The encoding of this bit is:

EPD1	Meaning
0	Perform translation table walks using TTBR1_EL2 .
1	A TLB miss on an address that is translated using TTBR1_EL2 generates a Translation fault. No translation table walk is performed.

A1, bit [22]

Selects whether [TTBR0_EL2](#) or [TTBR1_EL2](#) defines the ASID. The encoding of this bit is:

A1	Meaning
0	TTBR0_EL2 .ASID defines the ASID.
1	TTBR1_EL2 .ASID defines the ASID.

T1SZ, bits [21:16]

The size offset of the memory region addressed by [TTBR1_EL2](#). The region size is $2^{(64-T1SZ)}$ bytes.

The maximum and minimum possible values for T1SZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

TG0, bits [15:14]

Granule size for the [TTBR0_EL2](#).

TG0	Meaning
00	4KB
01	64KB
10	16KB

Other values are reserved.

If the value is programmed to either a reserved value, or a size that has not been implemented, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the sizes that has been implemented for all purposes other than the value read back from this register.

It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the value that corresponds to the size chosen.

SH0, bits [13:12]

Shareability attribute for memory associated with translation table walks using [TTBR0_EL2](#).

SH0	Meaning
00	Non-shareable
10	Outer Shareable
11	Inner Shareable

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in AArch64 System registers and translation table entries' in the ARM ARM, section K1.2.2.

ORGN0, bits [11:10]

Outer cacheability attribute for memory associated with translation table walks using [TTBR0_EL2](#).

ORGN0	Meaning
00	Normal memory, Outer Non-cacheable
01	Normal memory, Outer Write-Back Read-Allocate Write-Allocate Cacheable
10	Normal memory, Outer Write-Through Read-Allocate No Write-Allocate Cacheable
11	Normal memory, Outer Write-Back Read-Allocate No Write-Allocate Cacheable

IRGN0, bits [9:8]

Inner cacheability attribute for memory associated with translation table walks using [TTBR0_EL2](#).

IRGN0	Meaning
00	Normal memory, Inner Non-cacheable
01	Normal memory, Inner Write-Back Read-Allocate Write-Allocate Cacheable
10	Normal memory, Inner Write-Through Read-Allocate No Write-Allocate Cacheable
11	Normal memory, Inner Write-Back Read-Allocate No Write-Allocate Cacheable

EPD0, bit [7]

Translation table walk disable for translations using [TTBR0_EL2](#). This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using [TTBR0_EL2](#). The encoding of this bit is:

EPD0	Meaning
0	Perform translation table walks using TTBR0_EL2 .
1	A TLB miss on an address that is translated using TTBR0_EL2 generates a Translation fault. No translation table walk is performed.

Bit [6]

Reserved, RES0.

T0SZ, bits [5:0]

The size offset of the memory region addressed by [TTBR0_EL2](#). The region size is $2^{(64-T0SZ)}$ bytes.

The maximum and minimum possible values for T0SZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

Accessing the TCR_EL2

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
TCR_EL2	11	100	0010	0000	010
TCR_EL1	11	000	0010	0000	010

Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
TCR_EL2	x	x	0	-	-	n/a	RW
TCR_EL2	0	0	1	-	-	RW	RW
TCR_EL2	0	1	1	-	n/a	RW	RW
TCR_EL2	1	0	1	-	-	RW	RW
TCR_EL2	1	1	1	-	n/a	RW	RW
TCR_EL1	x	x	0	-	TCR_EL1	n/a	TCR_EL1
TCR_EL1	0	0	1	-	TCR_EL1	TCR_EL1	TCR_EL1
TCR_EL1	0	1	1	-	n/a	TCR_EL1	TCR_EL1
TCR_EL1	1	0	1	-	TCR_EL1	RW	TCR_EL1
TCR_EL1	1	1	1	-	n/a	RW	TCR_EL1

When [HCR_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic TCR_EL2 or TCR_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.NV](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

[SysReg_v83A_xml-00bet4](#)
[\(old\)](#)

htmldiff from-
SysReg_v83A_xml-00bet4

[\(new\)](#)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

VPIDR_EL2, Virtualization Processor ID Register

The VPIDR_EL2 characteristics are:

Purpose

Holds the value of the Virtualization Processor ID. This is the value returned by Non-secure EL1 reads of [MIDR_EL1](#).

This register is part of:

- The Virtualization registers functional group.
- The Identification registers functional group.

Configuration

AArch64 System register VPIDR_EL2 is architecturally mapped to AArch32 System register [VPIDR](#).

If EL2 is not implemented, reads of this register return the value of the [MIDR_EL1](#), and writes to the register are ignored.

RW fields in this register reset to architecturally UNKNOWN values.

Attributes

VPIDR_EL2 is a 32-bit register.

Field descriptions

The VPIDR_EL2 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Implementer								Variant				Architecture				PartNum								Revision							

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:

Hex representation	ASCII representation	Implementer
0x41	A	ARM Limited
0x42	B	Broadcom Corporation
0x43	C	Cavium Inc.
0x44	D	Digital Equipment Corporation
0x49	I	Infineon Technologies AG
0x4D	M	Motorola or Freescale Semiconductor Inc.
0x4E	N	NVIDIA Corporation
0x50	P	Applied Micro Circuits Corporation
0x51	Q	Qualcomm Inc.
0x56	V	Marvell International Ltd.
0x69	i	Intel Corporation

ARM can assign codes that are not published in this manual. All values not assigned by ARM are reserved and must not be used.

Variant, bits [23:20]

An IMPLEMENTATION DEFINED variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product.

Architecture, bits [19:16]

The permitted values of this field are:

Architecture	Meaning
0001	ARMv4
0010	ARMv4T
0011	ARMv5 (obsolete)
0100	ARMv5T
0101	ARMv5TE
0110	ARMv5TEJ
0111	ARMv6
1111	Architectural features are individually identified in the ID_* registers, see 'ID-identification registers, functional group' in the ARMv8 ARM, section K12.3.3G4.18.1.

All other values are reserved.

PartNum, bits [15:4]

An IMPLEMENTATION DEFINED primary part number for the device.

On processors implemented by ARM, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently.

Revision, bits [3:0]

An IMPLEMENTATION DEFINED revision number for the device.

Accessing the VPIDR_EL2

This register can be read using MRS with the following syntax:

```
MRS <Xt>, <systemreg>
```

This register can be written using MSR (register) with the following syntax:

```
MSR <systemreg>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
VPIDR_EL2	11	100	0000	0000	000

Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	RW
x	0	1	-	-	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when accessing this register.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2](#).NV==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2](#).NV==1, Non-secure accesses to this register from EL1 are trapped to EL2.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg v83A xml-00bet4](#)[\(old\)](#)

htmldiff from-

[\(new\)](#)[SysReg v83A xml-00bet4](#)[SysReg v83A xml-00bet5](#)

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

DC C1SW: Data or unified Cache line Clean and Invalidate by Set/Way

DC C1VAC: Data or unified Cache line Clean and Invalidate by VA to PoC

DC C2SW: Data or unified Cache line Clean by Set/Way

DC C2VAC: Data or unified Cache line Clean by VA to PoC

DC C2VAP: Data or unified Cache line Clean by VA to PoP

DC C2VAU: Data or unified Cache line Clean by VA to PoU

DC I1SW: Data or unified Cache line Invalidate by Set/Way

DC I1VAC: Data or unified Cache line Invalidate by VA to PoC

DC ZVA: Data Cache Zero by VA

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

S1_<op1>_<Cn>_<Cm>_<op2>: IMPLEMENTATION DEFINED maintenance instructions

TLBI ALLE1: TLB Invalidate All, EL1

TLBI ALLE1IS: TLB Invalidate All, EL1, Inner Shareable

TLBI ALLE2: TLB Invalidate All, EL2

TLBI ALLE2IS: TLB Invalidate All, EL2, Inner Shareable

TLBI ALLE3: TLB Invalidate All, EL3

TLBI ALLE3IS: TLB Invalidate All, EL3, Inner Shareable

[TLBI ASIDE1](#): TLB Invalidate by ASID, EL1

[TLBI ASIDE1IS](#): TLB Invalidate by ASID, EL1, Inner Shareable

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

TLBI IPAS2E1IS: TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

TLBI IPAS2LE1: TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1

TLBI IPAS2LE1IS: TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable

TLBI VAAE1: TLB Invalidate by VA, All ASID, EL1

TLBI VAAE1IS: TLB Invalidate by VA, All ASID, EL1, Inner Shareable

TLBI VAALE1: TLB Invalidate by VA, All ASID, Last level, EL1

TLBI VAALE1IS: TLB Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable

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

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

TLBI VAE2: TLB Invalidate by VA, EL2

TLBI VAE2IS: TLB Invalidate by VA, EL2, Inner Shareable

TLBI VAE3: TLB Invalidate by VA, EL3

TLBI VAE3IS: TLB Invalidate by VA, EL3, Inner Shareable

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

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

TLBI VALE2: TLB Invalidate by VA, Last level, EL2

TLBI VALE2IS: TLB Invalidate by VA, Last level, EL2, Inner Shareable

TLBI VALE3: TLB Invalidate by VA, Last level, EL3

TLBI VALE3IS: TLB Invalidate by VA, Last level, EL3, Inner Shareable

TLBI VMALLE1: TLB Invalidate by VMID, All at stage 1, EL1

TLBI VMALLE1IS: TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable

TLBI VMALLS12E1: TLB Invalidate by VMID, All at Stage 1 and 2, EL1

TLBI VMALLS12E1IS: TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner Shareable

28/09/2017 08:46:4140

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

TLBI ASIDE1, TLB Invalidate by ASID, EL1

The TLBI ASIDE1 characteristics are:

Purpose

Invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
 - Is from a level of lookup above the final level.
 - Is a non-global entry from the final level of lookup.
- If SCR_EL3.NS is 0, the entry would be required to translate an address using the Secure EL1&0 translation regime.
- If SCR_EL3.NS is 1, then:
 - If EL2 is not implemented, the entry would be required to translate an address using the Non-secure EL1&0 translation regime.
 - If EL2 is implemented and HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID, and would be required to translate an address using the Non-secure EL1&0 translation regime.
 - If EL2 is implemented and HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate an address using the EL2&0 translation regime.

The invalidation only applies to the PE that executes this instruction.

This System instruction is part of the TLB maintenance instructions functional group.

Configuration

There are no configuration notes.

Attributes

TLBI ASIDE1 is a 64-bit System instruction.

Field descriptions

The TLBI ASIDE1 input value bit assignments are:

63	62	61	60	59	58	57	56	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																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	

ASID, bits [63:48]

ASID value to match. Any appropriate TLB entries that match the ASID values will be affected by this operation.

If the implementation supports 16 bits of ASID, but only 8 bits are being used in the context being invalidated, the upper bits are RES0 and must be written to 0 by software performing the TLB maintenance.

Bits [47:0]

Reserved, RES0.

Executing the TLBI ASIDE1 instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2
ASIDE1	01	000	1000	0111	010

Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
0	0	1	-	WO	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	WO	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

When [HCR_EL2.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBI ASIDE1IS](#).

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when executing this System instruction.

When EL2 is implemented and is using AArch64 and $\text{SCR_EL3.NS}=1$ && $\text{HCR_EL2.E2H}=0$:

- If [HCR_EL2.TTLB](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch64 and $\text{SCR_EL3.NS}=1$ && $\text{HCR_EL2.E2H}=1$ && $\text{HCR_EL2.TGE}=0$:

- If [HCR_EL2.TTLB](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

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

The TLBI ASIDE1IS characteristics are:

Purpose

Invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
 - Is from a level of lookup above the final level.
 - Is a non-global entry from the final level of lookup.
- If SCR_EL3.NS is 0, the entry would be required to translate an address using the Secure EL1&0 translation regime.
- If SCR_EL3.NS is 1, then:
 - If EL2 is not implemented, the entry would be required to translate an address using the Non-secure EL1&0 translation regime.
 - If EL2 is implemented and HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID, and would be required to translate an address using the Non-secure EL1&0 translation regime.
 - If EL2 is implemented and HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate an address using the EL2&0 translation regime.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this instructions.

This System instruction is part of the TLB maintenance instructions functional group.

Configuration

There are no configuration notes.

Attributes

TLBI ASIDE1IS is a 64-bit System instruction.

Field descriptions

The TLBI ASIDE1IS input value bit assignments are:

63	62	61	60	59	58	57	56	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																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	

ASID, bits [63:48]

ASID value to match. Any appropriate TLB entries that match the ASID values will be affected by this operation.

If the implementation supports 16 bits of ASID, but only 8 bits are being used in the context being invalidated, the upper bits are RES0 and must be written to 0 by software performing the TLB maintenance.

Bits [47:0]

Reserved, RES0.

Executing the TLBI ASIDE1IS instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2
ASIDE1IS	01	000	1000	0011	010

Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
0	0	1	-	WO	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	WO	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

TLBI VAE1, TLB Invalidate by VA, EL1

The TLBI VAE1 characteristics are:

Purpose

Invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified address, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- If SCR_EL3.NS is 0, the entry would be required to translate using the Secure EL1&0 translation regime.
- If SCR_EL3.NS is 1, then:
 - If EL2 is not implemented, the entry would be required to translate using the Non-secure EL1&0 translation regime.
 - If EL2 is implemented and HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID, and would be required to translate using the Non-secure EL1&0 translation regime.
 - If EL2 is implemented and HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate using the EL2&0 translation regime.

The invalidation only applies to the PE that executes this instruction.

This System instruction is part of the TLB maintenance instructions functional group.

Configuration

There are no configuration notes.

Attributes

TLBI VAE1 is a 64-bit System instruction.

Field descriptions

The TLBI VAE1 input value bit assignments are:

63	62	61	60	59	58	57	56	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																0	0	0	0	VA[55:12]												
VA[55:12]																																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

Global TLB entries that match the VA value will be affected by this operation, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, but only 8 bits are being used in the context being invalidated, the upper bits are RES0, and must be written to 0 by software performing the TLB maintenance.

Bits [47:44]

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this operation.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAE1 instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2
VAE1	01	000	1000	0111	001

Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
0	0	1	-	WO	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	WO	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

When [HCR_EL2.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBI VAE1IS](#).

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

<u>SysReg_v83A_xml-00bet4</u> <u>(old)</u>	htmldiff from- SysReg_v83A_xml-00bet4	<u>(new)</u> <u>SysReg_v83A_xml-00bet5</u>
---	--	---

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

TLBI VAE1IS, TLB Invalidate by VA, EL1, Inner Shareable

The TLBI VAE1IS characteristics are:

Purpose

Invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified address, and one of the following applies:
 - The entry is from a level of lookup above the final level and matches the specified ASID.
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- If SCR_EL3.NS is 0, the entry would be required to translate using the Secure EL1&0 translation regime.
- If SCR_EL3.NS is 1, then:
 - If EL2 is not implemented, the entry would be required to translate using the Non-secure EL1&0 translation regime.
 - If EL2 is implemented and HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID, and would be required to translate using the Non-secure EL1&0 translation regime.
 - If EL2 is implemented and HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate using the EL2&0 translation regime.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this instructions.

This System instruction is part of the TLB maintenance instructions functional group.

Configuration

There are no configuration notes.

Attributes

TLBI VAE1IS is a 64-bit System instruction.

Field descriptions

The TLBI VAE1IS input value bit assignments are:

63	62	61	60	59	58	57	56	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																0	0	0	0	VA[55:12]															
VA[55:12]																																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

Global TLB entries that match the VA value will be affected by this operation, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, but only 8 bits are being used in the context being invalidated, the upper bits are RES0, and must be written to 0 by software performing the TLB maintenance.

Bits [47:44]

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this operation.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VAE1IS instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2
VAE1IS	01	000	1000	0011	001

Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
0	0	1	-	WO	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	WO	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.TTLB==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.TTLB==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

TLBI VALE1, TLB Invalidate by VA, Last level, EL1

The TLBI VALE1 characteristics are:

Purpose

Invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified address, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- If SCR_EL3.NS is 0, the entry would be required to translate using the Secure EL1&0 translation regime.
- If SCR_EL3.NS is 1, then:
 - If EL2 is not implemented, the entry would be required to translate using the Non-secure EL1&0 translation regime.
 - If EL2 is implemented and HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID, and would be required to translate using the Non-secure EL1&0 translation regime.
 - If EL2 is implemented and HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate using the EL2&0 translation regime.

The invalidation only applies to the PE that executes this instruction.

This System instruction is part of the TLB maintenance instructions functional group.

Configuration

There are no configuration notes.

Attributes

TLBI VALE1 is a 64-bit System instruction.

Field descriptions

The TLBI VALE1 input value bit assignments are:

63	62	61	60	59	58	57	56	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																0	0	0	0	VA[55:12]															
VA[55:12]																																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

Global TLB entries that match the VA value will be affected by this operation, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, but only 8 bits are being used in the context being invalidated, the upper bits are RES0, and must be written to 0 by software performing the TLB maintenance.

Bits [47:44]

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this operation.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VALE1 instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2
VALE1	01	000	1000	0111	101

Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
0	0	1	-	WO	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	WO	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

When [HCR_EL2.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBI VALE1IS](#).

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

<u>SysReg_v83A_xml-00bet4</u> <u>(old)</u>	htmldiff from-	<u>(new)</u> <u>SysReg_v83A_xml-00bet5</u>
---	----------------	---

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

TLBI VALE1IS, TLB Invalidate by VA, Last level, EL1, Inner Shareable

The TLBI VALE1IS characteristics are:

Purpose

Invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified address, and one of the following applies:
 - The entry is a global entry from the final level of lookup.
 - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- If SCR_EL3.NS is 0, the entry would be required to translate using the Secure EL1&0 translation regime.
- If SCR_EL3.NS is 1, then:
 - If EL2 is not implemented, the entry would be required to translate using the Non-secure EL1&0 translation regime.
 - If EL2 is implemented and HCR_EL2.{E2H, TGE} is not {1, 1}, the entry would be used with the current VMID, and would be required to translate using the Non-secure EL1&0 translation regime.
 - If EL2 is implemented and HCR_EL2.{E2H, TGE} is {1, 1}, the entry would be required to translate using the EL2&0 translation regime.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this instructions.

This System instruction is part of the TLB maintenance instructions functional group.

Configuration

There are no configuration notes.

Attributes

TLBI VALE1IS is a 64-bit System instruction.

Field descriptions

The TLBI VALE1IS input value bit assignments are:

63	62	61	60	59	58	57	56	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																0	0	0	0	VA[55:12]															
VA[55:12]																																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				

ASID, bits [63:48]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

Global TLB entries that match the VA value will be affected by this operation, regardless of the value of the ASID field.

If the implementation supports 16 bits of ASID, but only 8 bits are being used in the context being invalidated, the upper bits are RES0 and must be written to 0 by software performing the TLB maintenance.

Bits [47:44]

Reserved, RES0.

VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the ASID value (if appropriate) and VA will be affected by this operation.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

Executing the TLBI VALE1IS instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>, <Xt>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2
VALE1IS	01	000	1000	0011	101

Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
0	0	1	-	WO	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	WO	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*. Subject to the prioritization rules, the following traps and enables are applicable when executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==0 :

- If [HCR_EL2.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR_EL3.NS==1 && HCR_EL2.E2H==1 && HCR_EL2.TGE==0 :

- If [HCR_EL2.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)

htmldiff from-

[\(new\)](#)[\(old\)](#)[SysReg_v83A_xml-00bet4](#)[SysReg_v83A_xml-00bet5](#)

System Register index by instruction and encoding

Below are indexes for registers and operations accessed in the following ways:

For AArch32

- [MCR/MRC](#)[MRC/MCR](#)
- [MCCR/MRRCMRS/MSR](#)
- [MRS/MSR](#)[VMRS/VMSR](#)
- [VMRS/VMSRMRRM/MCCR](#)

For AArch64

- [ATMRS/MSR](#)
- [DCTLBI](#)
- [ICSYSL/SYS](#)
- [MRS/MSRDC/IC](#)
- [SYS/SYSLAT](#)
- [TLBI](#)

Registers and operations in AArch32

Accessed using [MRC/MCR/MRC](#):

Register selectors					Name	Description
opc1 coproc	opc2 opc1	CRn CRn	coproc CRm	CRm opc2		
000	000	0000	1110	0000	DBGDIDR	Debug ID Register
1110	000	0000	0000	000	DBGDIDR	Debug ID Register
111	000	0000	1110	0000	JIDR	Jazelle ID Register
1110	000	0000	0000	010	DBGDTRRXext	Debug OS Lock Data Transfer Register, Receive, External View
000	010	0000	1110	0000	DBGDTRRXext	Debug OS Lock Data Transfer Register, Receive, External View
1110	000	0000	0001	000	DBGDSCRint	Debug Status and Control Register, Internal View
000	000	0001	1110	0000	DBGDRAR	Debug ROM Address Register
1110	000	0000	0010	000	DBGDCCINT	DCC Interrupt Enable Register
111	000	0001	1110	0000	JOSCR	Jazelle OS Control Register
1110	000	0000	0010	010	DBGDSCRext	Debug Status and Control Register, External View
000	100	0001	1110	0000	DBGOSLAR	Debug OS Lock Access Register
1110	000	0000	0011	010	DBGDTRTXext	Debug OS Lock Data Transfer Register, Transmit
000	000	0010	1110	0000	DBGDSAR	Debug Self Address Register
1110	000	0000	0101	000	DBGDTRRXint	Debug Data Transfer Register, Receive

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRn	coproc CRm	CRm ope2		
111	000	0010	1110	0000	JMCR	Jazelle Main Configuration Register
1110	000	0000	0101	000	DBGDTRIXint	Debug Data Transfer Register, Transmit
000	111	0111	1110	0000	DBGDEVID2	Debug Device ID register 2
1110	000	0000	0110	000	DBGWFAR	Debug Watchpoint Fault Address Register
000	000	0000	1111	0000	MIDR	Main ID Register
1110	000	0000	0110	010	DBGOSECCR	Debug OS Lock Exception Catch Control Register
0011110	000	0000	11110111	0000000	CCSIDR DBGVCR	Current Debug Cache Vector Size ID Catch Register
0101110	000	0000	1111xxxx	0000100	CSSELR DBGBVR<n>	Cache Debug Size Breakpoint Selection Value Register
1001110	000	0000	1111xxxx	0000101	VPIDR DBGBCR<n>	Virtualization Debug Processor Breakpoint ID Control Register
000	001	0000	1111	0000	CTR	Cache Type Register
1110	000	0000	xxxx	110	DBGWVR<n>	Debug Watchpoint Value Registers
0011110	001000	0000	1111xxxx	0000111	CLIDR DBGWCR<n>	Cache Debug Level Watchpoint ID Control Register
000	010	0000	1111	0000	TCMTR	TCM Type Register
1110	000	0001	0000	000	DBGDRAR	Debug ROM Address Register
001	010	0000	1111	0000	CCSIDR2	Current Cache Size ID Register 2
1110	000	0001	0000	100	DBGOSLAR	Debug OS Lock Access Register
000	011	0000	1111	0000	TLBTR	TLB Type Register
1110	000	0001	0001	100	DBGOSLSR	Debug OS Lock Status Register
000	101	0000	1111	0000	MPIDR	Multiprocessor Affinity Register
1110	000	0001	0011	100	DBGOSDLR	Debug OS Double Lock Register
100	101	0000	1111	0000	VMPIDR	Virtualization Multiprocessor ID Register
1110	000	0001	0100	100	DBGPRCR	Debug Power Control Register
000	110	0000	1111	0000	REVIDR	Revision ID Register
1110	000	0001	xxxx	001	DBGBXVR<n>	Debug Breakpoint Extended Value Registers
001	111	0000	1111	0000	AIDR	Auxiliary ID Register
1110	000	0010	0000	000	DBGDSAR	Debug Self Address Register
000	000	0001	1111	0000	SCTLR	System Control Register
1110	000	0111	0000	111	DBGDEVID2	Debug Device ID register 2

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRm	coproc CRm	CRm ope2		
100	000	0001	1111	0000	HSCTLR	Hyp System Control Register
1110	000	0111	0001	111	DBGDEVID1	Debug Device ID register 1
000	001	0001	1111	0000	ACTLR	Auxiliary Control Register
1110	000	0111	0010	111	DBGDEVID	Debug Device ID register 0
1001110	001000	00010111	11111000	0000110	HACTLR DBGCLAIMSET	HypDebug Auxiliary Claim Control Tag Register Set register
000	010	0001	1111	0000	CPACR	Architectural Feature Access Control Register
1110	000	0111	1001	110	DBGCLAIMCLR	Debug Claim Tag Clear register
000	011	0001	1111	0000	ACTLR2	Auxiliary Control Register 2
1110	000	0111	1110	110	DBGAUTHSTATUS	Debug Authentication Status register
100	011	0001	1111	0000	HACTLR2	Hyp Auxiliary Control Register 2
1110	111	0000	0000	000	JIDR	Jazelle ID Register
000	000	0010	1111	0000	TTBR0	Translation Table Base Register 0
1110	111	0001	0000	000	JOSCR	Jazelle OS Control Register
000	001	0010	1111	0000	TTBR1	Translation Table Base Register 1
1110	111	0010	0000	000	JMCR	Jazelle Main Configuration Register
000	010	0010	1111	0000	TTBCR	Translation Table Base Control Register
1111	000	0000	0000	000	MIDR	Main ID Register
100	010	0010	1111	0000	HTCR	Hyp Translation Control Register
1111	000	0000	0000	001	CTR	Cache Type Register
000	011	0010	1111	0000	TTBCR2	Translation Table Base Control Register 2
1111	000	0000	0000	010	TCMTR	TCM Type Register
000	000	0011	1111	0000	DACR	Domain Access Control Register
1111	000	0000	0000	011	TLBTR	TLB Type Register
000	000	0101	1111	0000	DFSR	Data Fault Status Register
1111	000	0000	0000	101	MPIDR	Multiprocessor Affinity Register
000	001	0101	1111	0000	IFSR	Instruction Fault Status Register
1111	000	0000	0000	110	REVIDR	Revision ID Register
000	000	0110	1111	0000	DFAR	Data Fault Address Register
1111	000	0000	0001	000	ID_PFR0	Processor Feature Register 0
100	000	0110	1111	0000	HDFAR	Hyp Data Fault Address Register
1111	000	0000	0001	001	ID_PFR1	Processor Feature Register 1

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRn	coproc CRm	CRm ope2		
000	010	0110	1111	0000	IFAR	Instruction Fault Address Register
1111	000	0000	0001	010	ID_DFR0	Debug Feature Register 0
100	010	0110	1111	0000	HIFAR	Hyp Instruction Fault Address Register
1111	000	0000	0001	011	ID_AFR0	Auxiliary Feature Register 0
100	100	0110	1111	0000	HPFAR	Hyp IPA Fault Address Register
1111	000	0000	0001	100	ID_MMFR0	Memory Model Feature Register 0
100	001	1000	1111	0000	TLBIIPAS2IS	TLB Invalidate by Intermediate Physical Address, Stage 2, Inner Shareable
1111	000	0000	0001	101	ID_MMFR1	Memory Model Feature Register 1
100	101	1000	1111	0000	TLBIIPAS2LIS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, Inner Shareable
1111	000	0000	0001	110	ID_MMFR2	Memory Model Feature Register 2
000	000	1100	1111	0000	VBAR	Vector Base Address Register
1111	000	0000	0001	111	ID_MMFR3	Memory Model Feature Register 3
100	000	1100	1111	0000	HVBAR	Hyp Vector Base Address Register
1111	000	0000	0010	000	ID_ISAR0	Instruction Set Attribute Register 0
000	001	1100	1111	0000	MVBAR	Monitor Vector Base Address Register
1111	000	0000	0010	001	ID_ISAR1	Instruction Set Attribute Register 1
000	001	1100	1111	0000	RVBAR	Reset Vector Base Address Register
1111	000	0000	0010	010	ID_ISAR2	Instruction Set Attribute Register 2
000	010	1100	1111	0000	RMR	Reset Management Register
1111	000	0000	0010	011	ID_ISAR3	Instruction Set Attribute Register 3
100	010	1100	1111	0000	HRMR	Hyp Reset Management Register
1111	000	0000	0010	100	ID_ISAR4	Instruction Set Attribute Register 4
000	000	1101	1111	0000	FCSEIDR	FCSE Process ID register
1111	000	0000	0010	101	ID_ISAR5	Instruction Set Attribute Register 5
000	001	1101	1111	0000	CONTEXTIDR	Context ID Register
1111	000	0000	0010	110	ID_MMFR4	Memory Model Feature Register 4
000	010	1101	1111	0000	TPIDRURW	PL0 Read/Write Software Thread ID Register
1111	000	0000	0010	111	ID_ISAR6	Instruction Set Attribute Register 6

Register selectors					Name	Description
opc1 coproc	opc2 ope1	CRn CRm	coproc CRm	CRm ope2		
100	010	1101	1111	0000	HTPIDR	Hyp Software Thread ID Register
1111	000	0001	0000	000	SCTLR	System Control Register
000	011	1101	1111	0000	TPIDRURO	PL0 Read-Only Software Thread ID Register
1111	000	0001	0000	001	ACTLR	Auxiliary Control Register
000	100	1101	1111	0000	TPIDRPRW	PL1 Software Thread ID Register
1111	000	0001	0000	010	CPACR	Architectural Feature Access Control Register
000	000	1110	1111	0000	CNTFRQ	Counter-timer Frequency register
1111	000	0001	0000	011	ACTLR2	Auxiliary Control Register 2
000	000	0000	1110	0001	DBGDSCRint	Debug Status and Control Register, Internal View
1111	000	0001	0001	000	SCR	Secure Configuration Register
000	100	0001	1110	0001	DBGOSLSR	Debug OS Lock Status Register
1111	000	0001	0001	001	SDER	Secure Debug Enable Register
000	111	0111	1110	0001	DBGDEVID1	Debug Device ID register 1
1111	000	0001	0001	010	NSACR	Non-Secure Access Control Register
000	000	0000	1111	0001	ID_PFR0	Processor Feature Register 0
1111	000	0001	0011	001	SDCR	Secure Debug Control Register
000	001	0000	1111	0001	ID_PFR1	Processor Feature Register 1
1111	000	0010	0000	000	TTBR0	Translation Table Base Register 0
000	010	0000	1111	0001	ID_DFR0	Debug Feature Register 0
1111	000	0010	0000	001	TTBR1	Translation Table Base Register 1
000	011	0000	1111	0001	ID_AFR0	Auxiliary Feature Register 0
1111	000	0010	0000	010	TTBCR	Translation Table Base Control Register
000	100	0000	1111	0001	ID_MMFR0	Memory Model Feature Register 0
1111	000	0010	0000	011	TTBCR2	Translation Table Base Control Register 2
000	101	0000	1111	0001	ID_MMFR1	Memory Model Feature Register 1
1111	000	0011	0000	000	DACR	Domain Access Control Register
000	110	0000	1111	0001	ID_MMFR2	Memory Model Feature Register 2
1111	000	0100	0110	000	ICC_PMR	Interrupt Controller Interrupt Priority Mask Register
000	111	0000	1111	0001	ID_MMFR3	Memory Model Feature Register 3

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRm	coproc CRm	CRm ope2		
1111	000	0100	0110	000	ICV_PMR	Interrupt Controller Virtual-Interrupt Priority Mask Register
000	000	0001	1111	0001	SCR	Secure Configuration Register
1111	000	0101	0000	000	DFSR	Data Fault Status Register
1001111	000	00010101	11110000	0001001	HCR_IFSR	HypInstruction ConfigurationFault Status Register
000	001	0001	1111	0001	SDER	Secure Debug Enable Register
1111	000	0101	0001	000	ADFSR	Auxiliary Data Fault Status Register
100	001	0001	1111	0001	HDCCR	Hyp Debug Control Register
1111	000	0101	0001	001	AIFSR	Auxiliary Instruction Fault Status Register
000	010	0001	1111	0001	NSACR	Non-Secure Access Control Register
1111	000	0110	0000	000	DFAR	Data Fault Address Register
100	010	0001	1111	0001	HCPTR	Hyp Architectural Feature Trap Register
1111	000	0110	0000	010	IFAR	Instruction Fault Address Register
100	011	0001	1111	0001	HSTR	Hyp System Trap Register
1111	000	0111	0001	000	ICIALUIS	Instruction-Cache Invalidate All to PoU, Inner Shareable
100	100	0001	1111	0001	HCR2	Hyp Configuration Register 2
1111	000	0111	0001	110	BPIALLIS	Branch-Predictor Invalidate All, Inner Shareable
100	111	0001	1111	0001	HACR	Hyp Auxiliary Configuration Register
1111	000	0111	0100	000	PAR	Physical Address Register
100	010	0010	1111	0001	VTCT	Virtualization Translation Control Register
1111	000	0111	0101	000	ICIALLU	Instruction-Cache Invalidate All to PoU
000	000	0101	1111	0001	ADFSR	Auxiliary Data Fault Status Register
1111	000	0111	0101	001	ICIMVAU	Instruction-Cache line Invalidate by VA to PoU
100	000	0101	1111	0001	HADFSR	Hyp Auxiliary Data Fault Status Register
1111	000	0111	0101	100	CP15ISB	Instruction Synchronization Barrier System instruction
000	001	0101	1111	0001	AIFSR	Auxiliary Instruction Fault Status Register
1111	000	0111	0101	110	BPIALL	Branch-Predictor Invalidate All

Register selectors					Name	Description
opc1 copcode	opc2 opcode	CRn CRm	coproc CRm	CRm opcode		
100	001	0101	1111	0001	HAIFSR	Hyp Auxiliary Instruction Fault Status Register
1111	000	0111	0101	111	BPIMVA	Branch Predictor Invalidate by VA
000	000	0111	1111	0001	ICIALUIS	Instruction Cache Invalidate All to PoU, Inner Shareable
1111	000	0111	0110	001	DCIMVAC	Data Cache line Invalidate by VA to PoU
000	110	0111	1111	0001	BPIALLIS	Branch Predictor Invalidate All, Inner Shareable
1111	000	0111	0110	010	DCISW	Data Cache line Invalidate by Set/Way
000	000	1100	1111	0001	ISR	Interrupt Status Register
1111	000	0111	1000	000	ATS1CPR	Address Translate Stage 1 Current state PL1 Read
000	000	1110	1111	0001	CNTKCTL	Counter-timer Kernel Control register
1111	000	0111	1000	001	ATS1CPW	Address Translate Stage 1 Current state PL1 Write
1001111	000	11100111	11111000	0001010	CNTHCTL ATS1CUR	Counter-timerAddress HypTranslate ControlStage register1 Current state Unprivileged Read
000	000	0000	1110	0010	DBGDCCINT	DCC Interrupt Enable Register
1111	000	0111	1000	011	ATS1CUW	Address Translate Stage 1 Current state Unprivileged Write
000	010	0000	1110	0010	DBGDSCRExt	Debug Status and Control Register, External View
1111	000	0111	1000	100	ATS12NSOPR	Address Translate Stages 1 and 2 Non-secure Only PL1 Read
000	111	0111	1110	0010	DBGDEVID	Debug Device ID register 0
1111	000	0111	1000	101	ATS12NSOPW	Address Translate Stages 1 and 2 Non-secure Only PL1 Write
000	000	0000	1111	0010	ID_ISAR0	Instruction Set Attribute Register 0
1111	000	0111	1000	110	ATS12NSOUR	Address Translate Stages 1 and 2 Non-secure Only Unprivileged Read
000	001	0000	1111	0010	ID_ISAR1	Instruction Set Attribute Register 1
1111	000	0111	1000	111	ATS12NSOUW	Address Translate Stages 1 and 2 Non-secure Only Unprivileged Write
000	010	0000	1111	0010	ID_ISAR2	Instruction Set Attribute Register 2

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRm	coproc CRm	CRm ope2		
1111	000	0111	1001	000	ATS1CPRP	Address Translate Stage 1-Current state PL1 Read PAN
000	011	0000	1111	0010	ID_ISAR3	Instruction Set Attribute Register 3
1111	000	0111	1001	001	ATS1CPWP	Address Translate Stage 1-Current state PL1 Write PAN
000	100	0000	1111	0010	ID_ISAR4	Instruction Set Attribute Register 4
1111	000	0111	1010	001	DCCMVAC	Data Cache line Clean by VA to PoC
000	101	0000	1111	0010	ID_ISAR5	Instruction Set Attribute Register 5
1111	000	0111	1010	010	DCCSW	Data Cache line Clean by Set/Way
000	110	0000	1111	0010	ID_MMFR4	Memory Model Feature Register 4
1111	000	0111	1010	100	CP15DSB	Data Synchronization Barrier System instruction
000	111	0000	1111	0010	ID_ISAR6	Instruction Set Attribute Register 6
1111	000	0111	1010	101	CP15DMB	Data Memory Barrier System instruction
1001111	000	01010111	11111011	0010001	HSR DCCMVAU	HypData SyndromeCache Registerline Clean-by VA to PoU
000	000	1010	1111	0010	MAIR0	Memory Attribute Indirection Register 0
1111	000	0111	1110	001	DCCIMVAC	Data Cache line Clean and Invalidate by VA to PoC
000	000	1010	1111	0010	PRRR	Primary Region Remap Register
1111	000	0111	1110	010	DCCISW	Data Cache line Clean and Invalidate by Set/ Way
1001111	000	10101000	11110011	0010000	HMAIR0 TLBIALIS	HypTLB MemoryInvalidate AttributeAll, IndirectionInner Register 0Shareable
000	001	1010	1111	0010	MAIR1	Memory Attribute Indirection Register 1
1111	000	1000	0011	001	TLBIMVAIS	TLB Invalidate by VA, Inner-Shareable
000	001	1010	1111	0010	NMRR	Normal Memory Remap Register
1111	000	1000	0011	010	TLBIASIDIS	TLB Invalidate by ASID match, Inner Shareable
100	001	1010	1111	0010	HMAIR1	Hyp Memory Attribute Indirection Register 1
1111	000	1000	0011	011	TLBIMVAAIS	TLB Invalidate by VA, All-ASID, Inner Shareable
000	000	1110	1111	0010	CNTP_TVAL	Counter-timer Physical Timer TimerValue register

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRn	coproc CRm	CRm ope2		
1111	000	1000	0011	101	TLBIMVALIS	TLB Invalidate by VA, Last level, Inner Shareable
1001111	000	11101000	11110011	0010111	CNTHP_TVAL TLBIMVAALIS	Counter-timer TLB HypInvalidate Physical by Timer VA, Timer Value All register ASID, Last level Inner Shareable
000	001	1110	1111	0010	CNTP_CTL	Counter-timer Physical Timer Control register
1111	000	1000	0101	000	ITLBIALL	Instruction TLB Invalidate All
100	001	1110	1111	0010	CNTHP_CTL	Counter-timer Hyp Physical Timer Control register
1111	000	1000	0101	001	ITLBIMVA	Instruction TLB Invalidate by VA
000	010	0000	1110	0011	DBGDTRTXext	Debug OS Lock Data Transfer Register, Transmit
1111	000	1000	0101	010	ITLBIASID	Instruction TLB Invalidate by ASID match
000	100	0001	1110	0011	DBGOSDLR	Debug OS Double Lock Register
1111	000	1000	0110	000	DTLBIALL	Data TLB Invalidate All
000	001	0001	1111	0011	SDCR	Secure Debug Control Register
1111	000	1000	0110	001	DTLBIMVA	Data TLB Invalidate by VA
000	000	1000	1111	0011	TLBIALLIS	TLB Invalidate All, Inner Shareable
1111	000	1000	0110	010	DTLBIASID	Data TLB Invalidate by ASID match
1001111	000	1000	11110111	0011000	TLBIALLHIS TLBIALL	TLB Invalidate All, Hyp mode, Inner Shareable
000	001	1000	1111	0011	TLBIMVAIS	TLB Invalidate by VA, Inner Shareable
1111	000	1000	0111	001	TLBIMVA	TLB Invalidate by VA
1001111	001000	1000	11110111	0011010	TLBIMVAHIS TLBIASID	TLB Invalidate by VA, ASID Hyp mode, Inner Shareable match
000	010	1000	1111	0011	TLBIASIDIS	TLB Invalidate by ASID match, Inner Shareable
1111	000	1000	0111	011	TLBIMVAA	TLB Invalidate by VA, All ASID
000	011	1000	1111	0011	TLBIMVAAIS	TLB Invalidate by VA, All ASID, Inner Shareable
1111	000	1000	0111	101	TLBIMVAL	TLB Invalidate by VA, Last level
1001111	100000	1000	11110111	0011111	TLBIALLNSNHIS TLBIMVAAL	TLB Invalidate All by VA, Non-Secure All Non-Hyp ASID, Inner Last Shareable level
000	101	1000	1111	0011	TLBIMVALIS	TLB Invalidate by VA, Last level, Inner Shareable

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRn	coproc CRm	CRm ope2		
1111	000	1001	1100	000	PMCR	Performance Monitors Control Register
100	101	1000	1111	0011	TLBIMVALHIS	TLB Invalidate by VA, Last level, Hyp mode, Inner Shareable
1111	000	1001	1100	001	PMCNTENSET	Performance Monitors Count Enable Set register
000	111	1000	1111	0011	TLBIMVAALIS	TLB Invalidate by VA, All ASID, Last level, Inner Shareable
1111	000	1001	1100	010	PMCNTENCLR	Performance Monitors Count Enable Clear register
000	000	1010	1111	0011	AMAIR0	Auxiliary Memory Attribute Indirection Register 0
1111	000	1001	1100	011	PMOVSr	Performance Monitors Overflow Flag Status Register
100	000	1010	1111	0011	HAMAIR0	Hyp Auxiliary Memory Attribute Indirection Register 0
1111	000	1001	1100	100	PMSWINC	Performance Monitors Software Increment register
000	001	1010	1111	0011	AMAIR1	Auxiliary Memory Attribute Indirection Register 1
1111	000	1001	1100	101	PMSELR	Performance Monitors Event Counter Selection Register
100	001	1010	1111	0011	HAMAIR1	Hyp Auxiliary Memory Attribute Indirection Register 1
1111	000	1001	1100	110	PMCEID0	Performance Monitors Common Event Identification register 0
000	000	1110	1111	0011	CNTHV_TVAL	Counter-timer Virtual Timer TimerValue register (EL2)
1111	000	1001	1100	111	PMCEID1	Performance Monitors Common Event Identification register 1
000	000	1110	1111	0011	CNTV_TVAL	Counter-timer Virtual Timer TimerValue register
1111	000	1001	1101	000	PMCCNTR	Performance Monitors Cycle-Count Register
000	001	1110	1111	0011	CNTHV_CTL	Counter-timer Virtual Timer Control register (EL2)
1111	000	1001	1101	001	PMXEVTYPER	Performance Monitors Selected Event Type Register
000	001	1110	1111	0011	CNTV_CTL	Counter-timer Virtual Timer Control register
1111	000	1001	1101	010	PMXEVCNTR	Performance Monitors Selected Event Count Register

Register selectors					Name	Description
opc1 coproc	opc2 ope1	CRn CRm	coproc CRm	CRm ope2		
000	100	0001	1110	0100	DBGPRCR	Debug Power Control Register
1111	000	1001	1110	000	PMUSERENR	Performance Monitors User Enable Register
000	000	0111	1111	0100	PAR	Physical Address Register
1111	000	1001	1110	001	PMINTENSET	Performance Monitors Interrupt Enable Set register
100	001	1000	1111	0100	TLBIIPAS2	TLB Invalidate by Intermediate Physical Address, Stage 2
1111	000	1001	1110	010	PMINTENCLR	Performance Monitors Interrupt Enable Clear register
100	101	1000	1111	0100	TLBIIPAS2L	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level
1111	000	1001	1110	011	PMOVSSET	Performance Monitors Overflow Flag Status Set register
000	000	0000	1110	0101	DBGDTRRXint	Debug Data Transfer Register, Receive
1111	000	1001	1110	100	PMCEID2	Performance Monitors Common Event Identification register 2
000	000	0000	1110	0101	DBGDTRTXint	Debug Data Transfer Register, Transmit
1111	000	1001	1110	101	PMCEID3	Performance Monitors Common Event Identification register 3
01111111	000	01001010	11110010	0101000	DSPSR PRRR	Debug Primary Saved Region Program Status Remap Register
011	001	0100	1111	0101	DLR	Debug Link Register
1111	000	1010	0010	000	MAIR0	Memory Attribute Indirection Register 0
000	000	0111	1111	0101	ICIALLU	Instruction Cache Invalidate All to PoU
1111	000	1010	0010	001	NMRR	Normal Memory Remap Register
000	001	0111	1111	0101	ICIMVAU	Instruction Cache line Invalidate by VA to PoU
1111	000	1010	0010	001	MAIR1	Memory Attribute Indirection Register 1
000	100	0111	1111	0101	CP15ISB	Instruction Synchronization Barrier System instruction
1111	000	1010	0011	000	AMAIR0	Auxiliary Memory Attribute Indirection Register 0
000	110	0111	1111	0101	BPIALL	Branch Predictor Invalidate All
1111	000	1010	0011	001	AMAIR1	Auxiliary Memory Attribute Indirection Register 1
000	111	0111	1111	0101	BPIMVA	Branch Predictor Invalidate by VA
1111	000	1100	0000	000	VBAR	Vector Base-Address Register

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRn	coproc CRm	CRm ope2		
000	000	1000	1111	0101	ITLBIALL	Instruction TLB Invalidate All
1111	000	1100	0000	001	MVBAR	Monitor Vector Base Address Register
000	001	1000	1111	0101	ITLBIMVA	Instruction TLB Invalidate by VA
1111	000	1100	0000	001	RVBAR	Reset Vector Base Address Register
000	010	1000	1111	0101	ITLBIASID	Instruction TLB Invalidate by ASID match
1111	000	1100	0000	010	RMR	Reset Management Register
000	000	0000	1110	0110	DBGWFR	Debug Watchpoint Fault Address Register
1111	000	1100	0001	000	ISR	Interrupt Status Register
000	010	0000	1110	0110	DBGOSECRR	Debug OS Lock Exception Catch Control Register
1111	000	1100	1000	000	ICC_IAR0	Interrupt Controller Interrupt Acknowledge Register-0
000	000	0100	1111	0110	ICC_PMR	Interrupt Controller Interrupt Priority Mask Register
1111	000	1100	1000	000	ICV_IAR0	Interrupt Controller Virtual Interrupt Acknowledge Register-0
000	000	0100	1111	0110	ICV_PMR	Interrupt Controller Virtual Interrupt Priority Mask Register
1111	000	1100	1000	001	ICC_EOIR0	Interrupt Controller End Of Interrupt Register-0
000	001	0111	1111	0110	DCIMVAC	Data Cache line Invalidate by VA to PoC
1111	000	1100	1000	001	ICV_EOIR0	Interrupt Controller Virtual End Of Interrupt Register-0
000	010	0111	1111	0110	DCISW	Data Cache line Invalidate by Set/Way
1111	000	1100	1000	010	ICC_HPPIR0	Interrupt Controller Highest Priority Pending Interrupt Register-0
000	000	1000	1111	0110	DTLBIALL	Data TLB Invalidate All
1111	000	1100	1000	010	ICV_HPPIR0	Interrupt Controller Virtual Highest Priority Pending Interrupt Register-0
000	001	1000	1111	0110	DTLBIMVA	Data TLB Invalidate by VA
1111	000	1100	1000	011	ICC_BPR0	Interrupt Controller Binary Point Register-0
000	010	1000	1111	0110	DTLBIASID	Data TLB Invalidate by ASID match
1111	000	1100	1000	011	ICV_BPR0	Interrupt Controller Virtual Binary Point Register-0
000	000	0000	1110	0111	DBGVCR	Debug Vector Catch Register

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRn	coproc CRm	CRm ope2		
1111	000	1100	1000	1xx	ICC_AP0R<n>	Interrupt Controller Active Priorities Group 0 Registers
000	000	1000	1111	0111	TLBIALL	TLB Invalidate All
1111	000	1100	1000	1xx	ICV_AP0R<n>	Interrupt Controller Virtual Active Priorities Group 0 Registers
1001111	000	10001100	11111001	01110xx	TLBIALLH ICC_APIR<n>	TLBInterrupt InvalidateController All,Active HypPriorities modeGroup 1 Registers
000	001	1000	1111	0111	TLBIMVA	TLB Invalidate by VA
1111	000	1100	1001	0xx	ICV_APIR<n>	Interrupt Controller Virtual Active Priorities Group 1 Registers
100	001	1000	1111	0111	TLBIMVAH	TLB Invalidate by VA, Hyp mode
1111	000	1100	1011	001	ICC_DIR	Interrupt Controller Deactivate Interrupt Register
000	010	1000	1111	0111	TLBIASID	TLB Invalidate by ASID match
1111	000	1100	1011	001	ICV_DIR	Interrupt Controller Deactivate Virtual Interrupt Register
000	011	1000	1111	0111	TLBIMVAA	TLB Invalidate by VA, All ASID
1111	000	1100	1011	011	ICC_RPR	Interrupt Controller Running Priority Register
100	100	1000	1111	0111	TLBIALLSNH	TLB Invalidate All, Non-Secure Non-Hyp
1111	000	1100	1011	011	ICV_RPR	Interrupt Controller Virtual Running Priority Register
000	101	1000	1111	0111	TLBIMVAL	TLB Invalidate by VA, Last level
1111	000	1100	1100	000	ICC_IAR1	Interrupt Controller Interrupt Acknowledge Register 1
100	101	1000	1111	0111	TLBIMVALH	TLB Invalidate by VA, Last level, Hyp mode
1111	000	1100	1100	000	ICV_IAR1	Interrupt Controller Virtual Interrupt Acknowledge Register
000	111	1000	1111	0111	TLBIMVAAL	TLB Invalidate by VA, All ASID, Last level
1111	000	1100	1100	001	ICC_EOIR1	Interrupt Controller End Of Interrupt Register 1
000	110	0111	1110	1000	DBGCLAIMSET	Debug Claim Tag Set register
1111	000	1100	1100	001	ICV_EOIR1	Interrupt Controller Virtual End Of Interrupt Register 1
000	000	0111	1111	1000	ATS1CPR	Address Translate Stage 1 Current state PL1 Read
1111	000	1100	1100	010	ICC_HPIR1	Interrupt Controller Highest Priority Pending Interrupt Register 1

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRn	coproc CRm	CRm ope2		
1001111	000	01111100	11111100	1000010	ATS1HR ICV_HPPIR1	Address Interrupt Translate Controller Stage Virtual 1 Highest Hyp Priority mode Pending Read Interrupt Register
000	001	0111	1111	1000	ATS1CPW	Address Translate Stage 1 Current state PL1 Write
1111	000	1100	1100	011	ICC_BPR1	Interrupt Controller Binary Point Register 1
100	001	0111	1111	1000	ATS1HW	Address Translate Stage 1 Hyp mode Write
1111	000	1100	1100	011	ICV_BPR1	Interrupt Controller Virtual Binary Point Register 1
000	010	0111	1111	1000	ATS1CUR	Address Translate Stage 1 Current state Unprivileged Read
1111	000	1100	1100	100	ICC_CTLR	Interrupt Controller Control Register
000	011	0111	1111	1000	ATS1CUW	Address Translate Stage 1 Current state Unprivileged Write
1111	000	1100	1100	100	ICV_CTLR	Interrupt Controller Virtual Control Register
000	100	0111	1111	1000	ATS12NSOPR	Address Translate Stages 1 and 2 Non- secure Only PL1 Read
1111	000	1100	1100	101	ICC_SRE	Interrupt Controller System Register Enable register
000	101	0111	1111	1000	ATS12NSOPW	Address Translate Stages 1 and 2 Non- secure Only PL1 Write
1111	000	1100	1100	110	ICC_IGRPEN0	Interrupt Controller Interrupt Group 0 Enable register
000	110	0111	1111	1000	ATS12NSOUR	Address Translate Stages 1 and 2 Non- secure Only Unprivileged Read
1111	000	1100	1100	110	ICV_IGRPEN0	Interrupt Controller Virtual Interrupt Group 0 Enable register
000	111	0111	1111	1000	ATS12NSOUW	Address Translate Stages 1 and 2 Non- secure Only Unprivileged Write
1111	000	1100	1100	111	ICC_IGRPEN1	Interrupt Controller Interrupt Group 1 Enable register
000	000	1100	1111	1000	ICC_IAR0	Interrupt Controller Interrupt Acknowledge Register 0
1111	000	1100	1100	111	ICV_IGRPEN1	Interrupt Controller Virtual Interrupt Group 1 Enable register
000	000	1100	1111	1000	ICV_IAR0	Interrupt Controller Virtual Interrupt Acknowledge Register

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRm	coproc CRm	CRm ope2		
1111	000	1101	0000	000	FCSEIDR	FCSE Process ID register
000	001	1100	1111	1000	ICC_EOIR0	Interrupt Controller End Of Interrupt Register 0
1111	000	1101	0000	001	CONTEXTIDR	Context ID Register
000	001	1100	1111	1000	ICV_EOIR0	Interrupt Controller Virtual End Of Interrupt Register 0
1111	000	1101	0000	010	TPIDRURW	PL0 Read/Write Software Thread ID Register
000	010	1100	1111	1000	ICC_HPIR0	Interrupt Controller Highest Priority Pending Interrupt Register 0
1111	000	1101	0000	011	TPIDRURO	PL0 Read-Only Software Thread ID Register
000	010	1100	1111	1000	ICV_HPIR0	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
1111	000	1101	0000	100	TPIDRPRW	PL1 Software Thread ID Register
000	011	1100	1111	1000	ICC_BPR0	Interrupt Controller Binary Point Register 0
1111	000	1110	0000	000	CNTRQ	Counter-timer Frequency register
000	011	1100	1111	1000	ICV_BPR0	Interrupt Controller Virtual Binary Point Register 0
1111	000	1110	0001	000	CNTKCTL	Counter-timer Kernel Control register
100	0xx	1100	1111	1000	ICH_AP0R<n>	Interrupt Controller Hyp Active Priorities Group 0 Registers
1111	000	1110	0010	000	CNTP_TVAL	Counter-timer Physical Timer TimerValue register
000	1xx	1100	1111	1000	ICC_AP0R<n>	Interrupt Controller Active Priorities Group 0 Registers
1111	000	1110	0010	001	CNTP_CTL	Counter-timer Physical Timer Control register
000	1xx	1100	1111	1000	ICV_AP0R<n>	Interrupt Controller Virtual Active Priorities Group 0 Registers
1111	000	1110	0011	000	CNTV_TVAL	Counter-timer Virtual Timer TimerValue register
000	110	0111	1110	1001	DBGCLAIMCLR	Debug Claim Tag Clear register
1111	000	1110	0011	001	CNTV_CTL	Counter-timer Virtual Timer Control register
000	000	0111	1111	1001	ATS1CPRP	Address Translate Stage 1 Current state PL1 Read PAN
1111	000	1110	10xx	xxx	PMEVCNTR<n>	Performance Monitors Event Count Registers
000	001	0111	1111	1001	ATS1CPWP	Address Translate Stage 1 Current state PL1 Write PAN

Register selectors					Name	Description
opc1 cproce	opc2 ope1	CRn CRn	coproc CRm	CRm ope2		
1111	000	1110	1111	111	PMCCFILTR	Performance Monitors Cycle-Count Filter Register
100	101	1100	1111	1001	ICC_HSRE	Interrupt Controller Hyp System Register Enable register
1111	000	1110	11xx	xxx	PMEVTYPER<n>	Performance Monitors Event-Type Registers
000	0xx	1100	1111	1001	ICC_APIR<n>	Interrupt Controller Active Priorities Group 1 Registers
1111	001	0000	0000	000	CCSIDR	Current Cache Size ID Register
000	0xx	1100	1111	1001	ICV_APIR<n>	Interrupt Controller Virtual Active Priorities Group 1 Registers
1111	001	0000	0000	001	CLIDR	Cache Level ID Register
100	0xx	1100	1111	1001	ICH_APIR<n>	Interrupt Controller Hyp Active Priorities Group 1 Registers
1111	001	0000	0000	010	CCSIDR2	Current Cache Size ID Register 2
0001111	001	01110000	11110000	1010111	DCCMVAC AIDR	Data Auxiliary Cache ID line Clean by VA to PoC Register
000	010	0111	1111	1010	DCCSW	Data Cache line Clean by Set/Way
1111	010	0000	0000	000	CSSELR	Cache Size Selection Register
000	100	0111	1111	1010	CP15DSB	Data Synchronization Barrier System instruction
1111	011	0100	0101	000	DSPSR	Debug Saved Program Status Register
000	101	0111	1111	1010	CP15DMB	Data Memory Barrier System instruction
1111	011	0100	0101	001	DLR	Debug Link Register
000	001	0111	1111	1011	DCCMVAU	Data Cache line Clean by VA to PoU
1111	100	0000	0000	000	VPIDR	Virtualization Processor ID Register
100	000	1100	1111	1011	ICH_HCR	Interrupt Controller Hyp Control Register
1111	100	0000	0000	101	VMPIDR	Virtualization Multiprocessor ID Register
000	001	1100	1111	1011	ICC_DIR	Interrupt Controller Deactivate Interrupt Register
1111	100	0001	0000	000	HSTCLR	Hyp System Control Register
000	001	1100	1111	1011	ICV_DIR	Interrupt Controller Deactivate Virtual Interrupt Register
1111	100	0001	0000	001	HACTLR	Hyp Auxiliary Control Register
100	001	1100	1111	1011	ICH_VTR	Interrupt Controller VGIC Type Register
1111	100	0001	0000	011	HACTLR2	Hyp Auxiliary Control Register 2

Register selectors					Name	Description
opc1 copcode	opc2 ope1	CRn CRm	coproc CRm	CRm ope2		
100	010	1100	1111	1011	ICH_MISR	Interrupt Controller Maintenance Interrupt State Register
1111	100	0001	0001	000	HCR	Hyp Configuration Register
000	011	1100	1111	1011	ICC_RPR	Interrupt Controller Running Priority Register
1111	100	0001	0001	001	HDCCR	Hyp Debug Control Register
000	011	1100	1111	1011	ICV_RPR	Interrupt Controller Virtual Running Priority Register
1111	100	0001	0001	010	HCPTR	Hyp Architectural Feature Trap Register
100	011	1100	1111	1011	ICH_EISR	Interrupt Controller End of Interrupt Status Register
1111	100	0001	0001	011	HSTR	Hyp System Trap Register
100	101	1100	1111	1011	ICH_ELRSR	Interrupt Controller Empty List Register Status Register
1111	100	0001	0001	100	HCR2	Hyp Configuration Register 2
100	111	1100	1111	1011	ICH_VMCR	Interrupt Controller Virtual Machine Control Register
1111	100	0001	0001	111	HACR	Hyp Auxiliary Configuration Register
000	000	1001	1111	1100	PMCR	Performance Monitors Control Register
1111	100	0010	0000	010	HTCR	Hyp Translation Control Register
000	001	1001	1111	1100	PMCNTENSET	Performance Monitors Count Enable Set register
1111	100	0010	0001	010	VTCT	Virtualization Translation Control Register
000	010	1001	1111	1100	PMCNTENCLR	Performance Monitors Count Enable Clear register
1111	100	0101	0001	000	HADFSR	Hyp Auxiliary Data Fault Status Register
000	011	1001	1111	1100	PMOVS	Performance Monitors Overflow Flag Status Register
1111	100	0101	0001	001	HAIFSR	Hyp Auxiliary Instruction Fault Status Register
000	100	1001	1111	1100	PMSWINC	Performance Monitors Software Increment register
1111	100	0101	0010	000	HSR	Hyp Syndrome Register
000	101	1001	1111	1100	PMSELR	Performance Monitors Event Counter Selection Register
1111	100	0110	0000	000	HDFAR	Hyp Data Fault Address Register

Register selectors					Name	Description
opc1 copcode	opc2 ope1	CRn CRn	coproc CRm	CRm opc2		
000	110	1001	1111	1100	PMCEID0	Performance Monitors Common Event Identification register 0
1111	100	0110	0000	010	HIFAR	Hyp Instruction Fault Address Register
000	111	1001	1111	1100	PMCEID1	Performance Monitors Common Event Identification register 1
1111	100	0110	0000	100	HPFAR	Hyp IPA Fault Address Register
000	000	1100	1111	1100	ICC_IAR1	Interrupt Controller Interrupt Acknowledge Register 1
1111	100	0111	1000	000	ATS1HR	Address Translate Stage 1 Hyp mode Read
000	000	1100	1111	1100	ICV_IAR1	Interrupt Controller Virtual Interrupt Acknowledge Register
1111	100	0111	1000	001	ATS1HW	Address Translate Stage 1 Hyp mode Write
000	001	1100	1111	1100	ICC_EOIR1	Interrupt Controller End Of Interrupt Register 1
1111	100	1000	0000	001	TLBIIPAS2IS	TLB Invalidate by Intermediate Physical Address, Stage 2, Inner Shareable
000	001	1100	1111	1100	ICV_EOIR1	Interrupt Controller Virtual End Of Interrupt Register 1
1111	100	1000	0000	101	TLBIIPAS2LIS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, Inner Shareable
000	010	1100	1111	1100	ICC_HPIR1	Interrupt Controller Highest Priority Pending Interrupt Register 1
1111	100	1000	0011	000	TLBIALLHIS	TLB Invalidate All, Hyp mode, Inner Shareable
000	010	1100	1111	1100	ICV_HPIR1	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
1111	100	1000	0011	001	TLBIMVAHIS	TLB Invalidate by VA, Hyp mode, Inner Shareable
000	011	1100	1111	1100	ICC_BPR1	Interrupt Controller Binary Point Register 1
1111	100	1000	0011	100	TLBIALLNSNHIS	TLB Invalidate All, Non-Secure Non-Hyp, Inner Shareable
000	011	1100	1111	1100	ICV_BPR1	Interrupt Controller Virtual Binary Point Register 1
1111	100	1000	0011	101	TLBIMVALHIS	TLB Invalidate by VA, Last level, Hyp mode, Inner Shareable
0001111	100	11001000	11110100	1100001	ICC_CTLR TLBIIPAS2	InterruptTLB ControllerInvalidate Controlby RegisterIntermediate Physical Address, Stage 2

Register selectors					Name	Description
opc1 coproc	opc2 ope1	CRn CRn	coproc CRm	CRm ope2		
0001111	100	11001000	11110100	1100101	ICV_CTLR TLBIIPAS2L	InterruptTLB ControllerInvalidate Virtualby ControlIntermediate RegisterPhysical Address, Stage 2, Last level
1101111	100	11001000	11110111	1100000	ICC_MCTLR TLBIALH	InterruptTLB ControllerInvalidate MonitorAll, ControlHyp Registermode
000	101	1100	1111	1100	ICC_SRE	Interrupt Controller System Register Enable register
1111	100	1000	0111	001	TLBIMVAH	TLB Invalidate by VA, Hyp mode
110	101	1100	1111	1100	ICC_MSRE	Interrupt Controller Monitor System Register Enable register
1111	100	1000	0111	100	TLBIALNSNH	TLB Invalidate All, Non-Secure Non-Hyp
000	110	1100	1111	1100	ICC_IGRPEN0	Interrupt Controller Interrupt Group 0 Enable register
1111	100	1000	0111	101	TLBIMVALH	TLB Invalidate by VA, Last level, Hyp mode
000	110	1100	1111	1100	ICV_IGRPEN0	Interrupt Controller Virtual Interrupt Group 0 Enable register
1111	100	1010	0010	000	HMAIR0	Hyp Memory Attribute Indirection Register 0
000	111	1100	1111	1100	ICC_IGRPEN1	Interrupt Controller Interrupt Group 1 Enable register
1111	100	1010	0010	001	HMAIR1	Hyp Memory Attribute Indirection Register 1
000	111	1100	1111	1100	ICV_IGRPEN1	Interrupt Controller Virtual Interrupt Group 1 Enable register
1111	100	1010	0011	000	HAMAIR0	Hyp Auxiliary Memory Attribute Indirection Register 0
110	111	1100	1111	1100	ICC_MGRPEN1	Interrupt Controller Monitor Interrupt Group 1 Enable register
1111	100	1010	0011	001	HAMAIR1	Hyp Auxiliary Memory Attribute Indirection Register 1
000	000	1001	1111	1101	PMCCNTR	Performance Monitors Cycle Count Register
1111	100	1100	0000	000	HVBAR	Hyp Vector Base Address Register
000	001	1001	1111	1101	PMXEVTYPER	Performance Monitors Selected Event Type Register
1111	100	1100	0000	010	HRMR	Hyp Reset Management Register
000	010	1001	1111	1101	PMXVCNTR	Performance Monitors Selected Event Count Register

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRn	coproc CRm	CRm ope2		
1111	100	1100	1000	0xx	ICH_AP0R<n>	Interrupt Controller Hypervisor Active Priorities Group 0 Registers
0001111	110100	01111100	11101001	11100xx	DBGAUTHSTATUS ICH_APIR<n>	DebugInterrupt Authentication Controller StatusHyp registerActive Priorities Group 1 Registers
000	001	0111	1111	1110	DCCIMVAC	Data Cache line Clean and Invalidate by VA to PoC
1111	100	1100	1001	101	ICC_HSRE	Interrupt Controller Hypervisor System-Register Enable register
000	010	0111	1111	1110	DCCISW	Data Cache line Clean and Invalidate by Set/Way
1111	100	1100	1011	000	ICH_HCR	Interrupt Controller Hypervisor Control Register
000	000	1001	1111	1110	PMUSERENR	Performance Monitors User Enable Register
1111	100	1100	1011	001	ICH_VTR	Interrupt Controller VGIC Type Register
000	001	1001	1111	1110	PMINTENSET	Performance Monitors Interrupt Enable Set register
1111	100	1100	1011	010	ICH_MISR	Interrupt Controller Maintenance Interrupt State Register
000	010	1001	1111	1110	PMINTENCLR	Performance Monitors Interrupt Enable Clear register
1111	100	1100	1011	011	ICH_EISR	Interrupt Controller End of Interrupt Status Register
000	011	1001	1111	1110	PMOVSSET	Performance Monitors Overflow Flag Status Set register
1111	100	1100	1011	101	ICH_ELSR	Interrupt Controller Empty List Register Status Register
0001111	100	10011100	11111011	1110111	PMCEID2 ICH_VMCR	PerformanceInterrupt MonitorsController CommonVirtual EventMachine IdentificationControl register 2Register
000	101	1001	1111	1110	PMCEID3	Performance Monitors Common Event Identification register 3
1111	100	1100	110x	xxx	ICH_LR<n>	Interrupt Controller List Registers
000	111	1110	1111	1111	PMCCFILTR	Performance Monitors Cycle Count Filter Register
1111	100	1100	111x	xxx	ICH_LRC<n>	Interrupt Controller List Registers
000	xxx	1110	1111	10xx	PMEVCNTR<n>	Performance Monitors Event Count Registers
1111	100	1101	0000	010	HTPIDR	Hyp Software Thread ID Register

Register selectors					Name	Description
opc1 eoproc	opc2 ope1	CRn CRm	coproc CRm	CRm ope2		
100	xxx	1100	1111	110x	ICH_LR<n>	Interrupt Controller List Registers
1111	100	1110	0001	000	CNTHCTL	Counter-timer Hyp Control register
100	xxx	1100	1111	111x	ICH_LRC<n>	Interrupt Controller List Registers
1111	100	1110	0010	000	CNTHP_TVAL	Counter-timer Hyp Physical Timer Value register
0001111	xxx100	1110	11110010	11xx001	PMEVTYPEPER<n> CNTHP_CTL	Performance Counter-timer Monitors Hyp Event Physical Type Timer Registers Control register
000	100	0000	1110	xxxx	DBGBVR<n>	Debug Breakpoint Value Registers
1111	110	1100	1100	100	ICC_MCTLR	Interrupt Controller Monitor Control Register
000	101	0000	1110	xxxx	DBGBCR<n>	Debug Breakpoint Control Registers
1111	110	1100	1100	101	ICC_MSRE	Interrupt Controller Monitor System Register Enable register
0001111	110	00001100	11101100	xxxx111	DBGWVR<n> ICC_MGRPEN1	Debug Interrupt Watchpoint Controller Value Monitor Registers Interrupt Group 1 Enable register
000	111	0000	1110	xxxx	DBGWCR<n>	Debug Watchpoint Control Registers
000	001	0001	1110	xxxx	DBGBXVR<n>	Debug Breakpoint Extended Value Registers

Accessed using MCRR/MRRC:

Register selectors			Name	Description
opc1	coproc	CRm		
0000	1110	0001	DBGDRAR	Debug ROM Address Register
0000	1110	0010	DBGDSAR	Debug Self Address Register
0000	1111	0010	TTBR0	Translation Table Base Register 0
0001	1111	0010	TTBR1	Translation Table Base Register 1
0100	1111	0010	HTTBR	Hyp Translation Table Base Register
0110	1111	0010	VTTBR	Virtualization Translation Table Base Register
0000	1111	0111	PAR	Physical Address Register
0000	1111	1001	PMCCNTR	Performance Monitors Cycle Count Register
0000	1111	1100	ICC_SGI1R	Interrupt Controller Software Generated Interrupt Group 1 Register
0001	1111	1100	ICC_ASGI1R	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
0010	1111	1100	ICC_SGI0R	Interrupt Controller Software Generated Interrupt Group 0 Register
0000	1111	1110	CNTPCT	Counter-timer Physical Count register
0001	1111	1110	CNTVCT	Counter-timer Virtual Count register
0010	1111	1110	CNTP_CVAL	Counter-timer Physical Timer Compare Value register
0011	1111	1110	CNTHV_CVAL	Counter-timer Virtual Timer Compare Value register (EL2)
0011	1111	1110	CNTV_CVAL	Counter-timer Virtual Timer Compare Value register
0100	1111	1110	CNTVOFF	Counter-timer Virtual Offset register

Register selectors			Name	Description
opc1	coproc	CRm		
0110	1111	1110	CNTHP_CVAL	Counter-timer Hyp Physical Compare Value register

Accessed using MRS/MSR:

Register selectors			Name	Description
Rm	Mm1	M1R		
1	1	0000	SPSR_irq	Saved Program Status Register (IRQ mode)
0	1110	1	SPSR_fiq	Saved Program Status Register (FIQ mode)
1	1	0010	SPSR_svc	Saved Program Status Register (Supervisor mode)
1	0000	1	SPSR_irq	Saved Program Status Register (IRQ mode)
1	1	0100	SPSR_abt	Saved Program Status Register (Abort mode)
1	0010	1	SPSR_sve	Saved Program Status Register (Supervisor mode)
1	1	0110	SPSR_und	Saved Program Status Register (Undefined mode)
1	0100	1	SPSR_abt	Saved Program Status Register (Abort mode)
1	1	1100	SPSR_mon	Saved Program Status Register (Monitor mode)
1	0110	1	SPSR_und	Saved Program Status Register (Undefined mode)
1	01100	11101	SPSR_fiq SPSR_mon	Saved Program Status Register (FIQ Monitor mode)
0	1	1110	ELR_hyp	Exception Link Register (Hyp mode)
1	1110	0	ELR_hyp	Exception Link Register (Hyp mode)
1	1	1110	SPSR_hyp	Saved Program Status Register (Hyp mode)
1	1110	1	SPSR_hyp	Saved Program Status Register (Hyp mode)

Accessed using VMRS/VMSR:

Register selectors	Name	Description
regs pec-reg		
0000	FPSID	Floating-Point System ID register
0001	FPSCR	Floating-Point Status and Control Register
0101	MVFR2	Media and VFP Feature Register 2
0110	MVFR1	Media and VFP Feature Register 1
0111	MVFR0	Media and VFP Feature Register 0
1000	FPEXC	Floating-Point Exception Control register

Accessed using MRRR/MCRR:

Register selectors			Name	Description
coproc	opc1	CRm		
1110	0000	0001	DBGDRAR	Debug ROM Address Register
1110	0000	0010	DBGDSAR	Debug Self Address Register
1111	0000	0010	TTBR0	Translation Table Base Register 0
1111	0001	0010	TTBR1	Translation Table Base Register 1
1111	0100	0010	HTTBR	Hyp Translation Table Base Register
1111	0110	0010	VTTBR	Virtualization Translation Table Base Register
1111	0000	0111	PAR	Physical Address Register
1111	0000	1001	PMCCNTR	Performance Monitors Cycle Count Register
1111	0000	1100	ICC_SGHR	Interrupt Controller Software Generated Interrupt Group 1 Register
1111	0001	1100	ICC_ASGHR	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
1111	0010	1100	ICC_SGI0R	Interrupt Controller Software Generated Interrupt Group 0 Register

Register selectors			Name	Description
opproc	opel	CRm		
1111	0000	1110	CNTPCT	Counter-timer Physical Count register
1111	0001	1110	CNTVCT	Counter-timer Virtual Count register
1111	0010	1110	CNTP_CVAL	Counter-timer Physical Timer Compare Value register
1111	0011	1110	CNTV_CVAL	Counter-timer Virtual Timer Compare Value register
1111	0100	1110	CNTVOFF	Counter-timer Virtual Offset register
1111	0110	1110	CNTHP_CVAL	Counter-timer Hyp Physical Compare Value register

Registers and operations in AArch64

Accessed using **ATMRS/MSR**:

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
10	000	0001	0000	000	MDRAR_EL1	Monitor Debug ROM Address Register
10	000	0001	0000	100	OSLAR_EL1	OS Lock Access Register
10	000	0001	0001	100	OSLSR_EL1	OS Lock Status Register
10	000	0001	0011	100	OSDLR_EL1	OS Double Lock Register
10	000	0001	0100	100	DBGPRCR_EL1	Debug Power Control Register
10	000	0111	1000	110	DBGCLAIMSET_EL1	Debug Claim Tag Set register
10	000	0111	1001	110	DBGCLAIMCLR_EL1	Debug Claim Tag Clear register
10	000	0111	1110	110	DBGAUTHSTATUS_EL1	Debug Authentication Status register
11	000	0000	0000	000	MIDR_EL1	Main ID Register
11	001	0000	0000	000	CCSIDR_EL1	Current Cache Size ID Register
11	010	0000	0000	000	CSSELR_EL1	Cache Size Selection Register
11	100	0000	0000	000	VPIDR_EL2	Virtualization Processor ID Register
11	001	0000	0000	001	CLIDR_EL1	Cache Level ID Register
11	011	0000	0000	001	CTR_EL0	Cache Type Register
11	001	0000	0000	010	CCSIDR2_EL1	Current Cache Size ID Register 2
11	000	0000	0000	101	MPIDR_EL1	Multiprocessor Affinity Register
11	100	0000	0000	101	VMPIDR_EL2	Virtualization Multiprocessor ID Register
11	000	0000	0000	110	REVIDR_EL1	Revision ID Register
11	001	0000	0000	111	AIDR_EL1	Auxiliary ID Register
11	011	0000	0000	111	DCZID_EL0	Data Cache Zero ID register
11	000	0000	0001	000	ID_PFR0_EL1	AArch32 Processor Feature Register 0
11	000	0000	0001	001	ID_PFR1_EL1	AArch32 Processor Feature Register 1

op0	op1	Register selectors		op2	Name	Description
CRn	CRm					
11	000	0000	0001	010	ID_DFR0_EL1	AArch32 Debug Feature Register 0
11	000	0000	0001	011	ID_AFR0_EL1	AArch32 Auxiliary Feature Register 0
11	000	0000	0001	100	ID_MMFR0_EL1	AArch32 Memory Model Feature Register 0
11	000	0000	0001	101	ID_MMFR1_EL1	AArch32 Memory Model Feature Register 1
11	000	0000	0001	110	ID_MMFR2_EL1	AArch32 Memory Model Feature Register 2
11	000	0000	0001	111	ID_MMFR3_EL1	AArch32 Memory Model Feature Register 3
11	000	0000	0010	000	ID_ISAR0_EL1	AArch32 Instruction Set Attribute Register 0
11	000	0000	0010	001	ID_ISAR1_EL1	AArch32 Instruction Set Attribute Register 1
11	000	0000	0010	010	ID_ISAR2_EL1	AArch32 Instruction Set Attribute Register 2
11	000	0000	0010	011	ID_ISAR3_EL1	AArch32 Instruction Set Attribute Register 3
11	000	0000	0010	100	ID_ISAR4_EL1	AArch32 Instruction Set Attribute Register 4
11	000	0000	0010	101	ID_ISAR5_EL1	AArch32 Instruction Set Attribute Register 5
11	000	0000	0010	110	ID_MMFR4_EL1	AArch32 Memory Model Feature Register 4
11	000	0000	0010	111	ID_ISAR6_EL1	AArch32 Instruction Set Attribute Register 6
11	000	0000	0011	000	MVFR0_EL1	AArch32 Media and VFP Feature Register 0
11	000	0000	0011	001	MVFR1_EL1	AArch32 Media and VFP Feature Register 1
11	000	0000	0011	010	MVFR2_EL1	AArch32 Media and VFP Feature Register 2
11	000	0000	0100	000	ID_AA64PFR0_EL1	AArch64 Processor Feature Register 0
11	000	0000	0100	001	ID_AA64PFR1_EL1	AArch64 Processor Feature Register 1
11	000	0000	0101	000	ID_AA64DFR0_EL1	AArch64 Debug Feature Register 0
11	000	0000	0101	001	ID_AA64DFR1_EL1	AArch64 Debug Feature Register 1
11	000	0000	0101	100	ID_AA64AFR0_EL1	AArch64 Auxiliary Feature Register 0
11	000	0000	0101	101	ID_AA64AFR1_EL1	AArch64 Auxiliary Feature Register 1

op0	op1	Register selectors		op2	Name	Description
CRn	CRm					
11	000	0000	0110	000	ID_AA64ISAR0_EL1	AArch64 Instruction Set Attribute Register 0
11	000	0000	0110	001	ID_AA64ISAR1_EL1	AArch64 Instruction Set Attribute Register 1
11	000	0000	0111	000	ID_AA64MMFR0_EL1	AArch64 Memory Model Feature Register 0
11	000	0000	0111	001	ID_AA64MMFR1_EL1	AArch64 Memory Model Feature Register 1
11	000	0000	0111	010	ID_AA64MMFR2_EL1	AArch64 Memory Model Feature Register 2
11	000	0001	0000	000	SCTLR_EL1	System Control Register (EL1)
11	100	0001	0000	000	SCTLR_EL2	System Control Register (EL2)
11	110	0001	0000	000	SCTLR_EL3	System Control Register (EL3)
11	000	0001	0000	001	ACTLR_EL1	Auxiliary Control Register (EL1)
11	100	0001	0000	001	ACTLR_EL2	Auxiliary Control Register (EL2)
11	110	0001	0000	001	ACTLR_EL3	Auxiliary Control Register (EL3)
11	000	0001	0000	010	CPACR_EL1	Architectural Feature Access Control Register
11	100	0001	0001	000	HCR_EL2	Hypervisor Configuration Register
11	110	0001	0001	000	SCR_EL3	Secure Configuration Register
11	100	0001	0001	001	MDCR_EL2	Monitor Debug Configuration Register (EL2)
11	110	0001	0001	001	SDER32_EL3	AArch32 Secure Debug Enable Register
11	100	0001	0001	010	CPTR_EL2	Architectural Feature Trap Register (EL2)
11	110	0001	0001	010	CPTR_EL3	Architectural Feature Trap Register (EL3)
11	100	0001	0001	011	HSTR_EL2	Hypervisor System Trap Register
11	100	0001	0001	111	HACR_EL2	Hypervisor Auxiliary Control Register
11	110	0001	0011	001	MDCR_EL3	Monitor Debug Configuration Register (EL3)
11	000	0010	0000	000	TTBR0_EL1	Translation Table Base Register 0 (EL1)
11	100	0010	0000	000	TTBR0_EL2	Translation Table Base Register 0 (EL2)
11	110	0010	0000	000	TTBR0_EL3	Translation Table Base Register 0 (EL3)

op0	op1	Register selectors		op2	Name	Description
CRn	CRm					
11	000	0010	0000	001	TTBR1_EL1	Translation Table Base Register 1 (EL1)
11	100	0010	0000	001	TTBR1_EL2	Translation Table Base Register 1 (EL2)
11	000	0010	0000	010	TCR_EL1	Translation Control Register (EL1)
11	100	0010	0000	010	TCR_EL2	Translation Control Register (EL2)
11	110	0010	0000	010	TCR_EL3	Translation Control Register (EL3)
11	000	0010	0001	000	APIAKeyLo_EL1	Pointer Authentication Key A for Instruction (bits[63:0])
11	100	0010	0001	000	VTBR_EL2	Virtualization Translation Table Base Register
11	000	0010	0001	001	APIAKeyHi_EL1	Pointer Authentication Key A for Instruction (bits[127:64])
11	000	0010	0001	010	APIBKeyLo_EL1	Pointer Authentication Key B for Instruction (bits[63:0])
11	100	0010	0001	010	VTCR_EL2	Virtualization Translation Control Register
11	000	0010	0001	011	APIBKeyHi_EL1	Pointer Authentication Key B for Instruction (bits[127:64])
11	000	0010	0010	000	APDAKeyLo_EL1	Pointer Authentication Key A for Data (bits[63:0])
11	000	0010	0010	001	APDAKeyHi_EL1	Pointer Authentication Key A for Data (bits[127:64])
11	000	0010	0010	010	APDBKeyLo_EL1	Pointer Authentication Key B for Data (bits[63:0])
11	000	0010	0010	011	APDBKeyHi_EL1	Pointer Authentication Key B for Data (bits[127:64])
11	000	0010	0011	000	APGAKeyLo_EL1	Pointer Authentication Key A for Code (bits[63:0])
11	000	0010	0011	001	APGAKeyHi_EL1	Pointer Authentication Key A for Code (bits[127:64])
11	100	0011	0000	000	DACR32_EL2	Domain Access Control Register
11	000	0100	0000	000	SPSR_EL1	Saved Program Status Register (EL1)

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
11	100	0100	0000	000	SPSR_EL2	Saved Program Status Register (EL2)
11	110	0100	0000	000	SPSR_EL3	Saved Program Status Register (EL3)
11	000	0100	0000	001	ELR_EL1	Exception Link Register (EL1)
11	100	0100	0000	001	ELR_EL2	Exception Link Register (EL2)
11	110	0100	0000	001	ELR_EL3	Exception Link Register (EL3)
11	000	0100	0001	000	SP_EL0	Stack Pointer (EL0)
11	100	0100	0001	000	SP_EL1	Stack Pointer (EL1)
11	110	0100	0001	000	SP_EL2	Stack Pointer (EL2)
11	000	0100	0010	000	SPSel	Stack Pointer Select
11	011	0100	0010	000	NZCV	Condition Flags
11	011	0100	0010	001	DAIF	Interrupt Mask Bits
11	000	0100	0010	010	CurrentEL	Current Exception Level
11	000	0100	0010	011	PAN	Privileged Access Never
11	000	0100	0010	100	UAO	User Access Override
11	100	0100	0011	000	SPSR_irq	Saved Program Status Register (IRQ mode)
11	100	0100	0011	001	SPSR_abt	Saved Program Status Register (Abort mode)
11	100	0100	0011	010	SPSR_und	Saved Program Status Register (Undefined mode)
11	100	0100	0011	011	SPSR_fiq	Saved Program Status Register (FIQ mode)
11	011	0100	0100	000	FPCR	Floating-point Control Register
11	011	0100	0100	001	FPSR	Floating-point Status Register
11	011	0100	0101	000	DSPSR_EL0	Debug Saved Program Status Register
11	011	0100	0101	001	DLR_EL0	Debug Link Register
11	000	0100	0110	000	ICC_PMR_EL1	Interrupt Controller Interrupt Priority Mask Register
11	000	0100	0110	000	ICV_PMR_EL1	Interrupt Controller Virtual Interrupt Priority Mask Register
11	100	0101	0000	001	IFSR32_EL2	Instruction Fault Status Register (EL2)
11	000	0101	0001	000	AFSR0_EL1	Auxiliary Fault Status Register 0 (EL1)
11	100	0101	0001	000	AFSR0_EL2	Auxiliary Fault Status Register 0 (EL2)
11	110	0101	0001	000	AFSR0_EL3	Auxiliary Fault Status Register 0 (EL3)

op0	op1	Register selectors		op2	Name	Description
CRn	CRm					
11	000	0101	0001	001	AFSR1_EL1	Auxiliary Fault Status Register 1 (EL1)
11	100	0101	0001	001	AFSR1_EL2	Auxiliary Fault Status Register 1 (EL2)
11	110	0101	0001	001	AFSR1_EL3	Auxiliary Fault Status Register 1 (EL3)
11	000	0101	0010	000	ESR_EL1	Exception Syndrome Register (EL1)
11	100	0101	0010	000	ESR_EL2	Exception Syndrome Register (EL2)
11	110	0101	0010	000	ESR_EL3	Exception Syndrome Register (EL3)
11	100	0101	0011	000	FPEXC32_EL2	Floating Point Exception Control register
11	000	0110	0000	000	FAR_EL1	Fault Address Register (EL1)
11	100	0110	0000	000	FAR_EL2	Fault Address Register (EL2)
11	110	0110	0000	000	FAR_EL3	Fault Address Register (EL3)
11	100	0110	0000	100	HPFAR_EL2	Hypervisor IPA Fault Address Register
11	000	0111	0100	000	PAR_EL1	Physical Address Register
11	011	1001	1100	000	PMCR_EL0	Performance Monitors Control Register
11	011	1001	1100	001	PMCNTENSET_EL0	Performance Monitors Count Enable Set register
11	011	1001	1100	010	PMCNTENCLR_EL0	Performance Monitors Count Enable Clear register
11	011	1001	1100	011	PMOVSCLR_EL0	Performance Monitors Overflow Flag Status Clear Register
11	011	1001	1100	100	PMSWINC_EL0	Performance Monitors Software Increment register
11	011	1001	1100	101	PMSELR_EL0	Performance Monitors Event Counter Selection Register
11	011	1001	1100	110	PMCEID0_EL0	Performance Monitors Common Event Identification register 0
11	011	1001	1100	111	PMCEID1_EL0	Performance Monitors Common Event Identification register 1
11	011	1001	1101	000	PMCCNTR_EL0	Performance Monitors Cycle Count Register
11	011	1001	1101	001	PMXEVTYPER_EL0	Performance Monitors Selected Event Type Register

op0	op1	Register selectors		op2	Name	Description
CRn	CRm					
11	011	1001	1101	010	PMXEVCNTR_EL0	Performance Monitors Selected Event Count Register
11	011	1001	1110	000	PMUSERENR_EL0	Performance Monitors User Enable Register
11	000	1001	1110	001	PMINTENSET_EL1	Performance Monitors Interrupt Enable Set-register
11	000	1001	1110	010	PMINTENCLR_EL1	Performance Monitors Interrupt Enable Clear-register
11	011	1001	1110	011	PMOVSSET_EL0	Performance Monitors Overflow Flag Status Set register
11	000	1010	0010	000	MAIR_EL1	Memory Attribute Indirection Register (EL1)
11	100	1010	0010	000	MAIR_EL2	Memory Attribute Indirection Register (EL2)
11	110	1010	0010	000	MAIR_EL3	Memory Attribute Indirection Register (EL3)
11	000	1010	0011	000	AMAIR_EL1	Auxiliary Memory Attribute Indirection Register (EL1)
11	100	1010	0011	000	AMAIR_EL2	Auxiliary Memory Attribute Indirection Register (EL2)
11	110	1010	0011	000	AMAIR_EL3	Auxiliary Memory Attribute Indirection Register (EL3)
11	000	1010	0100	000	LORSA_EL1	LORegion Start Address (EL1)
11	000	1010	0100	001	LOREA_EL1	LORegion End Address (EL1)
11	000	1010	0100	010	LORN_EL1	LORegion Number (EL1)
11	000	1010	0100	011	LORC_EL1	LORegion Control (EL1)
11	000	1010	0100	111	LORID_EL1	LORegionID (EL1)
11	000	1100	0000	000	VBAR_EL1	Vector Base Address Register (EL1)
11	100	1100	0000	000	VBAR_EL2	Vector Base Address Register (EL2)
11	110	1100	0000	000	VBAR_EL3	Vector Base Address Register (EL3)
11	000	1100	0000	001	RVBAR_EL1	Reset Vector Base Address Register (if EL2 and EL3 not implemented)
11	100	1100	0000	001	RVBAR_EL2	Reset Vector Base Address Register (if EL3 not implemented)
11	110	1100	0000	001	RVBAR_EL3	Reset Vector Base Address Register (if EL3 implemented)
11	000	1100	0000	010	RMR_EL1	Reset Management Register (EL1)

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
11	100	1100	0000	010	RMR_EL2	Reset Management Register (EL2)
11	110	1100	0000	010	RMR_EL3	Reset Management Register (EL3)
11	000	1100	0001	000	ISR_EL1	Interrupt Status Register
11	000	1100	1000	000	ICC_IAR0_EL1	Interrupt Controller Interrupt Acknowledge Register 0
11	000	1100	1000	000	ICV_IAR0_EL1	Interrupt Controller Virtual Interrupt Acknowledge Register 0
11	000	1100	1000	001	ICC_EOIR0_EL1	Interrupt Controller End Of Interrupt Register 0
11	000	1100	1000	001	ICV_EOIR0_EL1	Interrupt Controller Virtual End Of Interrupt Register 0
11	000	1100	1000	010	ICC_HPPIR0_EL1	Interrupt Controller Highest Priority Pending Interrupt Register 0
11	000	1100	1000	010	ICV_HPPIR0_EL1	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
11	000	1100	1000	011	ICC_BPR0_EL1	Interrupt Controller Binary Point Register 0
11	000	1100	1000	011	ICV_BPR0_EL1	Interrupt Controller Virtual Binary Point Register 0
11	100	1100	1000	0xx	ICH_AP0R<n>_EL2	Interrupt Controller Hyp Active Priorities Group 0 Registers
11	000	1100	1000	1xx	ICC_AP0R<n>_EL1	Interrupt Controller Active Priorities Group 0 Registers
11	000	1100	1000	1xx	ICV_AP0R<n>_EL1	Interrupt Controller Virtual Active Priorities Group 0 Registers
11	000	1100	1001	0xx	ICC_APIR<n>_EL1	Interrupt Controller Active Priorities Group 1 Registers
11	000	1100	1001	0xx	ICV_APIR<n>_EL1	Interrupt Controller Virtual Active Priorities Group 1 Registers
11	100	1100	1001	0xx	ICH_APIR<n>_EL2	Interrupt Controller Hyp Active Priorities Group 1 Registers
11	100	1100	1001	101	ICC_SRE_EL2	Interrupt Controller System Register Enable register (EL2)
11	100	1100	1011	000	ICH_HCR_EL2	Interrupt Controller Hyp Control Register
11	000	1100	1011	001	ICC_DIR_EL1	Interrupt Controller Deactivate Interrupt Register

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
11	000	1100	1011	001	ICV_DIR_EL1	Interrupt Controller Deactivate Virtual Interrupt Register
11	100	1100	1011	001	ICH_VTR_EL2	Interrupt Controller VGIC Type Register
11	100	1100	1011	010	ICH_MISR_EL2	Interrupt Controller Maintenance Interrupt State Register
11	000	1100	1011	011	ICC_RPR_EL1	Interrupt Controller Running Priority Register
11	000	1100	1011	011	ICV_RPR_EL1	Interrupt Controller Virtual Running Priority Register
11	100	1100	1011	011	ICH_EISR_EL2	Interrupt Controller End of Interrupt Status Register
11	000	1100	1011	101	ICC_SGI1R_EL1	Interrupt Controller Software Generated Interrupt Group 1 Register
11	100	1100	1011	101	ICH_ELRSR_EL2	Interrupt Controller Empty List Register Status Register
11	000	1100	1011	110	ICC_ASGI1R_EL1	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
11	000	1100	1011	111	ICC_SGI0R_EL1	Interrupt Controller Software Generated Interrupt Group 0 Register
11	100	1100	1011	111	ICH_VMCR_EL2	Interrupt Controller Virtual Machine Control Register
11	000	1100	1100	000	ICC_IAR1_EL1	Interrupt Controller Interrupt Acknowledge Register 1
11	000	1100	1100	000	ICV_IAR1_EL1	Interrupt Controller Virtual Interrupt Acknowledge Register 1
11	000	1100	1100	001	ICC_EOIR1_EL1	Interrupt Controller End Of Interrupt Register 1
11	000	1100	1100	001	ICV_EOIR1_EL1	Interrupt Controller Virtual End Of Interrupt Register 1
11	000	1100	1100	010	ICC_HPIR1_EL1	Interrupt Controller Highest Priority Pending Interrupt Register 1
11	000	1100	1100	010	ICV_HPIR1_EL1	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
11	000	1100	1100	011	ICC_BPR1_EL1	Interrupt Controller Binary Point Register 1

op0	op1	Register selectors		op2	Name	Description
CRn	CRm					
11	000	1100	1100	011	ICV_BPR1_EL1	Interrupt Controller Virtual Binary Point Register 1
11	000	1100	1100	100	ICC_CTLR_EL1	Interrupt Controller Control Register (EL1)
11	000	1100	1100	100	ICV_CTLR_EL1	Interrupt Controller Virtual Control Register
11	110	1100	1100	100	ICC_CTLR_EL3	Interrupt Controller Control Register (EL3)
11	000	1100	1100	101	ICC_SRE_EL1	Interrupt Controller System Register Enable register (EL1)
11	110	1100	1100	101	ICC_SRE_EL3	Interrupt Controller System Register Enable register (EL3)
11	000	1100	1100	110	ICC_IGRPEN0_EL1	Interrupt Controller Interrupt Group 0 Enable register
11	000	1100	1100	110	ICV_IGRPEN0_EL1	Interrupt Controller Virtual Interrupt Group 0 Enable register
11	000	1100	1100	111	ICC_IGRPEN1_EL1	Interrupt Controller Interrupt Group 1 Enable register
11	000	1100	1100	111	ICV_IGRPEN1_EL1	Interrupt Controller Virtual Interrupt Group 1 Enable register
11	110	1100	1100	111	ICC_IGRPEN1_EL3	Interrupt Controller Interrupt Group 1 Enable register (EL3)
11	100	1100	110x	xxx	ICH_LR<n>_EL2	Interrupt Controller List Registers
11	000	1101	0000	001	CONTEXTIDR_EL1	Context ID Register (EL1)
11	100	1101	0000	001	CONTEXTIDR_EL2	Context ID Register (EL2)
11	011	1101	0000	010	TPIDR_EL0	EL0 Read/Write Software Thread ID Register
11	100	1101	0000	010	TPIDR_EL2	EL2 Software Thread ID Register
11	110	1101	0000	010	TPIDR_EL3	EL3 Software Thread ID Register
11	011	1101	0000	011	TPIDRRO_EL0	EL0 Read-Only Software Thread ID Register
11	000	1101	0000	100	TPIDR_EL1	EL1 Software Thread ID Register
11	011	1110	0000	000	CNTFRQ_EL0	Counter-timer Frequency register
11	011	1110	0000	001	CNTPCT_EL0	Counter-timer Physical Count register
11	011	1110	0000	010	CNTVCT_EL0	Counter-timer Virtual Count register
11	100	1110	0000	011	CNTVOFF_EL2	Counter-timer Virtual Offset register

op0	op1	Register selectors		op2	Name	Description
CRn	CRm					
11	000	1110	0001	000	CNTKCTL_EL1	Counter-timer Kernel Control register
11	100	1110	0001	000	CNTHCTL_EL2	Counter-timer Hypervisor Control register
11	011	1110	0010	000	CNTP_TVAL_EL0	Counter-timer Physical Timer TimerValue register
11	100	1110	0010	000	CNTHP_TVAL_EL2	Counter-timer Hypervisor Physical Timer TimerValue register
11	111	1110	0010	000	CNTPS_TVAL_EL1	Counter-timer Physical Secure Timer TimerValue register
11	011	1110	0010	001	CNTP_CTL_EL0	Counter-timer Physical Timer Control register
11	100	1110	0010	001	CNTHP_CTL_EL2	Counter-timer Hypervisor Physical Timer Control register
11	111	1110	0010	001	CNTPS_CTL_EL1	Counter-timer Physical Secure Timer Control register
11	011	1110	0010	010	CNTP_CVAL_EL0	Counter-timer Physical Timer CompareValue register
11	100	1110	0010	010	CNTHP_CVAL_EL2	Counter-timer Hypervisor Physical Timer CompareValue register
11	111	1110	0010	010	CNTPS_CVAL_EL1	Counter-timer Physical Secure Timer CompareValue register
11	011	1110	0011	000	CNTV_TVAL_EL0	Counter-timer Virtual Timer TimerValue register
11	100	1110	0011	000	CNTHV_TVAL_EL2	Counter-timer Hypervisor Virtual Timer TimerValue register (EL2)
11	011	1110	0011	001	CNTV_CTL_EL0	Counter-timer Virtual Timer Control register
11	100	1110	0011	001	CNTHV_CTL_EL2	Counter-timer Hypervisor Virtual Timer Control register (EL2)
11	011	1110	0011	010	CNTV_CVAL_EL0	Counter-timer Virtual Timer CompareValue register
11	100	1110	0011	010	CNTHV_CVAL_EL2	Counter-timer Hypervisor Virtual Timer CompareValue register (EL2)
11	011	1110	10xx	xxx	PMEVCNTR<n>_EL0	Performance Monitors Event Count Registers
11	011	1110	1111	111	PMCCFILTR_EL0	Performance Monitors Cycle Count Filter Register

op0	op1	Register selectors		op2	Name	Description
CRn	CRm					
11	011	1110	11xx	xxx	PMEVTYPEPER<n>_EL0	Performance Monitors Event Type Registers
11	xxx	1x11	xxxx	xxx	S3_<op1>_<Cn>_<Cm>_<op2>	IMPLEMENTATION DEFINED registers
0110	000	01110000	10000000	000010	AT S1E1R OSDTRRX_EL1	AddressOS TranslateLock StageData 1Transfer EL1Register, ReadReceive
0110	100011	01110000	10000001	000	AT S1E2R MDCCSR_EL0	AddressMonitor TranslateDCC StageStatus 1 EL2 ReadRegister
01	110	0111	1000	000	AT S1E3R	Address Translate Stage 1 EL3 Read
10	000	0000	0010	000	MDCCINT_EL1	Monitor DCC Interrupt Enable Register
0110	000	01110000	10010010	000010	AT S1E1RP MDSCR_EL1	AddressMonitor TranslateDebug StageSystem 1Control EL1 Read PANRegister
0110	000	01110000	10000011	001010	AT S1E1W OSDTRTX_EL1	AddressOS TranslateLock StageData 1Transfer EL1Register, WriteTransmit
0110	100011	01110000	10000100	001000	AT S1E2W DBGDTR_EL0	AddressDebug TranslateData StageTransfer 1Register, EL2 Writehalf-duplex
0110	110011	01110000	10000101	001000	AT S1E3W DBGDTRRX_EL0	AddressDebug TranslateData StageTransfer 1Register, EL3 WriteReceive
01	000	0111	1001	001	AT S1E1WP	Address Translate Stage 1 EL1 Write PAN
10	011	0000	0101	000	DBGDTRTX_EL0	Debug Data Transfer Register, Transmit
0110	000	01110000	10000110	010	AT S1E0R OSECCR_EL1	AddressOS TranslateLock StageException 1Catch EL0Control ReadRegister
01	000	0111	1000	011	AT S1E0W	Address Translate Stage 1 EL0 Write
10	100	0000	0111	000	DBGVCR32_EL2	Debug Vector Catch Register
01	100	0111	1000	100	AT S12E1R	Address Translate Stages 1 and 2 EL1 Read
10	000	0000	xxxx	100	DBGBVR<n>_EL1	Debug Breakpoint Value Registers
0110	100000	01110000	1000xxxx	101	AT S12E1W DBGBCR<n>_EL1	AddressDebug TranslateBreakpoint StagesControl 1 and 2 EL1 WriteRegister

Register selectors		Register selectors		Name	Description
op0	op1	CRn	CRm		
0110	100000	01110000	1000xxxx	110	AT S12E0R DBGWVR<n>_EL1 Address Debug Translate Watchpoint Stages Value 1 and 2 EL0 Read Registers
0110	100000	01110000	1000xxxx	111	AT S12E0W DBGWCR<n>_EL1 Address Debug Translate Watchpoint Stages Control 1 and 2 EL0 Write Register

Accessed using DCTLBI:

Register selectors		Register selectors		Name	Description
op0	op1	CRn	CRm		
01	000	1000	0011	011	TLBI VAAEHIS TLB Invalidate by VA, All ASID, EL1, Inner Shareable
01	100	1000	0011	100	TLBI ALLEHS TLB Invalidate All, EL1, Inner Shareable
01	000	1000	0011	101	TLBI VALEHS TLB Invalidate by VA, Last level, EL1, Inner Shareable
01	100	1000	0011	101	TLBI VALE2IS TLB Invalidate by VA, Last level, EL2, Inner Shareable
01	110	1000	0011	101	TLBI VALE3IS TLB Invalidate by VA, Last level, EL3, Inner Shareable
01	100	1000	0011	110	TLBI VMALLS12EHIS TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner Shareable
01	000	1000	0011	111	TLBI VAALEHIS TLB Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable
01	100	1000	0100	001	TLBI IPAS2E1 TLB Invalidate by Intermediate Physical Address, Stage 2, EL1
01	100	1000	0100	101	TLBI IPAS2LE1 TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
01	000	1000	0111	000	TLBI VMALLE1 TLB Invalidate by VMID, All at stage 1, EL1
01	100	1000	0111	000	TLBI ALLE2 TLB Invalidate All, EL2
01	110	1000	0111	000	TLBI ALLE3 TLB Invalidate All, EL3
01	000	1000	0111	001	TLBI VAE1 TLB Invalidate by VA, EL1
01	100	1000	0111	001	TLBI VAE2 TLB Invalidate by VA, EL2
01	110	1000	0111	001	TLBI VAE3 TLB Invalidate by VA, EL3
01	000	1000	0111	010	TLBI ASIDE1 TLB Invalidate by ASID, EL1
01	000	1000	0111	011	TLBI VAAE1 TLB Invalidate by VA, All ASID, EL1
01	100	1000	0111	100	TLBI ALLE1 TLB Invalidate All, EL1
01	000	1000	0111	101	TLBI VALE1 TLB Invalidate by VA, Last level, EL1
01	100	1000	0111	101	TLBI VALE2 TLB Invalidate by VA, Last level, EL2
01	110	1000	0111	101	TLBI VALE3 TLB Invalidate by VA, Last level, EL3
01	100	1000	0111	110	TLBI VMALLS12E1 TLB Invalidate by VMID, All at Stage 1 and 2, EL1
01	000	1000	0111	111	TLBI VAALE1 TLB Invalidate by VA, All ASID, Last level, EL1
01	011100	01111000	01000000	001	DC ZVA TLBI IPAS2EHIS Data TLB Cache Invalidate Zero by VA Intermediate Physical

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			Address, Stage 2, EL1, Inner Shareable
01	000100	01111000	01100000	001101	DC IVAC TLBI IPAS2LE1IS	DataTLB orInvalidate unifiedby CacheIntermediate linePhysical InvalidateAddress, byStage VA2, toLast PoClevel, EL1, Inner Shareable
01	011000	01111000	10100011	001000	DC CVAC TLBI VMALLE1IS	DataTLB orInvalidate unifiedby CacheVMID, lineAll Cleanat bystage VA1, toEL1, PoCInner Shareable
01	011100	01111000	10110011	001000	DC CVAU TLBI ALLE2IS	DataTLB orInvalidate unifiedAll, CacheEL2, lineInner Clean by VA to PoUShareable
01	011110	01111000	11000011	001000	DC CVAP TLBI ALLE3IS	DataTLB orInvalidate unifiedAll, CacheEL3, lineInner Clean by VA to PoPShareable
01	011000	01111000	11100011	001	DC CIVAC TLBI VAE1IS	DataTLB or unified Cache line Clean and Invalidate by VA, toEL1, PoCInner Shareable
01	000100	01111000	01100011	010001	DC ISW TLBI VAE2IS	DataTLB orInvalidate unifiedby CacheVA, lineEL2, InvalidateInner by Set/WayShareable
01	000110	01111000	10100011	010001	DC CSW TLBI VAE3IS	DataTLB orInvalidate unifiedby CacheVA, lineEL3, CleanInner by Set/WayShareable
01	000	01111000	11100011	010	DC CISW TLBI ASIDE1IS	DataTLB orInvalidate unifiedby CacheASID, lineEL1, CleanInner and Invalidate by Set/WayShareable

Accessed using **ICSYSL/SYS**:

op0	op1	Register selectors		op2	Rt	Name	
		CRn	CRm				
01xxx	0111x11	0111xxxx	0101xxx	001	-	IC IVAU S1 <op1> <Cn> <Cm> <op2>	Instruct CacheD linemai VA to P
01	000	0111	0001	000	11111	IC IALLUIS	Instruct to PoU,
01	000	0111	0101	000	11111	IC IALLU	Instruct to PoU

Accessed using **MRSDC/MSRIC**:

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
1101	000	00000111	00000001	000	MIDR_EL1 IC IALLUIS	MainInstruction IDCache RegisterInvalidate A to PoU, Inner Share
11	001	0000	0000	000	CCSIDR_EL1	Current Cache Size Register
01	011	0111	0100	001	DC ZVA	Data Cache Zero by

op0	op1	Register selectors		op2	Name	Description
CRn	CRm					
11	010	0000	0000	000	CSSELR_EL1	Cache Size Selection Register
01	000	0111	0101	000	ICIALLU	Instruction Cache Invalidate All to PoU
1101	100011	00000111	00000101	000001	VPIDR_EL2 IC IVAU	Virtualization Instruction Processor Cache ID Register Invalidate by VA to PoU
1001	000	00010111	00000110	000001	MDRAR_EL1 DC IVAC	Monitor Data Debug ROM unified Address Cache Registerline Invalidate by VA to PoC
1101	000	00010111	00000110	000010	SCTLR_EL1 DC ISW	SystemData Control Register unified (EL1) Cache line Invalidate by Set/W
1101	100011	00010111	00001010	000001	SCTLR_EL2 DC CVAC	SystemData Control Register unified (EL2) Cache line Clean by VA to PoC
11	110	0001	0000	000	SCTLR_EL3	System Control Register (EL3)
01	000	0111	1010	010	DC CSW	Data or unified Cache line Clean by Set/W
1101	000011	00100111	00001011	000001	TTBR0_EL1 DC CVAU	TranslationData Table Base unified Register Cache 0line (EL1) Clean by VA to PoU
1101	100011	00100111	00001100	000001	TTBR0_EL2 DC CVAP	TranslationData Table Base unified Register Cache 0line (EL2) Clean by VA to PoP
1101	110011	00100111	00001110	000001	TTBR0_EL3 DC CIVAC	TranslationData Table Base unified Register Cache 0line (EL3) Clean and Invalidate by VA to PoU
11	100	0011	0000	000	DACR32_EL2	Domain Access Control Register
01	000	0111	1110	010	DC CISW	Data or unified Cache line Clean and Invalidate by Set/W
11	000	0100	0000	000	SPSR_EL1	Saved Program Status Register (EL1)
11	100	0100	0000	000	SPSR_EL2	Saved Program Status Register (EL2)
11	110	0100	0000	000	SPSR_EL3	Saved Program Status Register (EL3)
11	000	0110	0000	000	FAR_EL1	Fault Address Register (EL1)
11	100	0110	0000	000	FAR_EL2	Fault Address Register (EL2)
11	110	0110	0000	000	FAR_EL3	Fault Address Register (EL3)
11	000	1100	0000	000	VBAR_EL1	Vector Base Address Register (EL1)
11	100	1100	0000	000	PMSCR_EL2	Statistical Profiling Control Register (EL2)

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
11	100	1100	0000	000	VBAR_EL2	Vector Base Address Register (EL2)
11	110	1100	0000	000	VBAR_EL3	Vector Base Address Register (EL3)
11	011	1110	0000	000	CNTFRQ_EL0	Counter-timer Frequency register
11	000	0000	0001	000	ID_PFR0_EL1	AArch32 Processor Feature Register 0
10	011	0000	0001	000	MDCCSR_EL0	Monitor DCC Status Register
11	100	0001	0001	000	HCR_EL2	Hypervisor Configuration Register
11	110	0001	0001	000	SCR_EL3	Secure Configuration Register
11	000	0010	0001	000	APIAKeyLo_EL1	Pointer Authentication Key A for Instructions (bits[63:0])
11	100	0010	0001	000	VTBR_EL2	Virtualization Translation Table Base Register
11	000	0100	0001	000	SP_EL0	Stack Pointer (EL0)
11	100	0100	0001	000	SP_EL1	Stack Pointer (EL1)
11	110	0100	0001	000	SP_EL2	Stack Pointer (EL2)
11	000	0101	0001	000	AFSR0_EL1	Auxiliary Fault Status Register 0 (EL1)
11	100	0101	0001	000	AFSR0_EL2	Auxiliary Fault Status Register 0 (EL2)
11	110	0101	0001	000	AFSR0_EL3	Auxiliary Fault Status Register 0 (EL3)
11	000	1100	0001	000	ISR_EL1	Interrupt Status Register
11	000	1110	0001	000	CNTKCTL_EL1	Counter-timer Kernel Control register
11	100	1110	0001	000	CNTHCTL_EL2	Counter-timer Hypervisor Control register
10	000	0000	0010	000	MDCCINT_EL1	Monitor DCC Interrupt Enable Register
11	000	0000	0010	000	ID_ISAR0_EL1	AArch32 Instruction Set Attribute Register
11	000	0010	0010	000	APDAKeyLo_EL1	Pointer Authentication Key A for Data (bits[63:0])
11	000	0100	0010	000	SPSel	Stack Pointer Select
11	011	0100	0010	000	NZCV	Condition Flags
11	000	0101	0010	000	ESR_EL1	Exception Syndrome Register (EL1)
11	100	0101	0010	000	ESR_EL2	Exception Syndrome Register (EL2)
11	110	0101	0010	000	ESR_EL3	Exception Syndrome Register (EL3)
11	000	1010	0010	000	MAIR_EL1	Memory Attribute Indirection Register (EL1)
11	100	1010	0010	000	MAIR_EL2	Memory Attribute Indirection Register (EL2)
11	110	1010	0010	000	MAIR_EL3	Memory Attribute Indirection Register (EL3)

op0	op1	Register selectors		op2	Name	Description
CRn	CRm					
11	011	1110	0010	000	CNTP_TVAL_EL0	Counter-timer Physical Timer TimerValue register
11	100	1110	0010	000	CNTHP_TVAL_EL2	Counter-timer Hypervisor Physical Timer TimerValue register
11	111	1110	0010	000	CNTPS_TVAL_EL1	Counter-timer Physical Secure Timer TimerValue register
11	000	0000	0011	000	MVFR0_EL1	AArch32 Media and VFP Feature Register 0
11	000	0010	0011	000	APGAKeyLo_EL1	Pointer Authentication Key A for Code (bits[63:0])
11	100	0100	0011	000	SPSR_irq	Saved Program Status Register (IRQ mode)
11	100	0101	0011	000	FPEXC32_EL2	Floating-Point Exception Control register
11	000	1010	0011	000	AMAIR_EL1	Auxiliary Memory Attribute Indirection Register (EL1)
11	100	1010	0011	000	AMAIR_EL2	Auxiliary Memory Attribute Indirection Register (EL2)
11	110	1010	0011	000	AMAIR_EL3	Auxiliary Memory Attribute Indirection Register (EL3)
11	011	1110	0011	000	CNTV_TVAL_EL0	Counter-timer Virtual Timer TimerValue register
11	100	1110	0011	000	CNTHV_TVAL_EL2	Counter-timer Virtual Timer TimerValue register (EL2)
11	000	0000	0100	000	ID_AA64PFR0_EL1	AArch64 Processor Feature Register 0
10	011	0000	0100	000	DBGDTR_EL0	Debug Data Transfer Register, half-duplex
11	011	0100	0100	000	FPCR	Floating-point Control Register
11	000	0111	0100	000	PAR_EL1	Physical Address Register
11	000	1010	0100	000	LORSA_EL1	LORegion Start Address (EL1)
11	000	0000	0101	000	ID_AA64DFR0_EL1	AArch64 Debug Feature Register 0
10	011	0000	0101	000	DBGDTRRX_EL0	Debug Data Transfer Register, Receive
10	011	0000	0101	000	DBGDTRTX_EL0	Debug Data Transfer Register, Transmit
11	011	0100	0101	000	DSPSR_EL0	Debug Saved Program Status Register
11	000	0000	0110	000	ID_AA64ISAR0_EL1	AArch64 Instruction Set Attribute Register 0
11	000	0100	0110	000	ICC_PMR_EL1	Interrupt Controller Interrupt Priority Mask Register
11	000	0100	0110	000	ICV_PMR_EL1	Interrupt Controller Virtual Interrupt Priority Mask Register

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
11	000	0000	0111	000	ID_AA64MMFR0_EL1	AArch64 Memory Model Feature Register 0
10	100	0000	0111	000	DBGVCR32_EL2	Debug Vector Catch Register
11	000	1100	1000	000	ICC_IAR0_EL1	Interrupt Controller Interrupt Acknowledge Register 0
11	000	1100	1000	000	ICV_IAR0_EL1	Interrupt Controller Virtual Interrupt Acknowledge Register 0
11	000	1001	1001	000	PMSCR_EL1	Statistical Profiling Control Register (EL1)
11	000	1001	1010	000	PMBLIMITR_EL1	Profiling Buffer Limit Address Register
11	000	1001	1010	000	PMSIRR_EL1	Sampling Interval Reload Register
11	100	1100	1011	000	ICH_HCR_EL2	Interrupt Controller Hyp Control Register
11	011	1001	1100	000	PMCR_EL0	Performance Monitor Control Register
11	000	1100	1100	000	ICC_IAR1_EL1	Interrupt Controller Interrupt Acknowledge Register 1
11	000	1100	1100	000	ICV_IAR1_EL1	Interrupt Controller Virtual Interrupt Acknowledge Register 1
11	011	1001	1101	000	PMCCNTR_EL0	Performance Monitor Cycle Count Register
11	011	1001	1110	000	PMUSERENR_EL0	Performance Monitor User Enable Register
11	001	0000	0000	001	CLIDR_EL1	Cache Level ID Register
11	011	0000	0000	001	CTR_EL0	Cache Type Register
11	000	0001	0000	001	ACTLR_EL1	Auxiliary Control Register (EL1)
11	100	0001	0000	001	ACTLR_EL2	Auxiliary Control Register (EL2)
11	110	0001	0000	001	ACTLR_EL3	Auxiliary Control Register (EL3)
11	000	0010	0000	001	TTBR1_EL1	Translation Table Base Register 1 (EL1)
11	100	0010	0000	001	TTBR1_EL2	Translation Table Base Register 1 (EL2)
11	000	0100	0000	001	ELR_EL1	Exception Link Register (EL1)
11	100	0100	0000	001	ELR_EL2	Exception Link Register (EL2)
11	110	0100	0000	001	ELR_EL3	Exception Link Register (EL3)
11	100	0101	0000	001	IFSR32_EL2	Instruction Fault Status Register (EL2)
11	000	1100	0000	001	RVBAR_EL1	Reset Vector Base Address Register (if EL2 and EL3 not implemented)
11	100	1100	0000	001	RVBAR_EL2	Reset Vector Base Address Register (if EL3 not implemented)

op0	op1	Register selectors		op2	Name	Description
CRn	CRm					
11	110	1100	0000	001	RVBAR_EL3	Reset Vector Base Address Register (if EL3 implemented)
11	000	1101	0000	001	CONTEXTIDR_EL1	Context ID Register (EL1)
11	100	1101	0000	001	CONTEXTIDR_EL2	Context ID Register (EL2)
11	011	1110	0000	001	CNTPCT_EL0	Counter-timer Physical Count register
11	000	0000	0001	001	ID_PFR1_EL1	AArch32 Processor Feature Register 1
11	100	0001	0001	001	MDCR_EL2	Monitor Debug Configuration Register (EL2)
11	110	0001	0001	001	SDER32_EL3	AArch32 Secure Debug Enable Register
11	000	0010	0001	001	APIAKeyHi_EL1	Pointer Authentication Key A for Instructions (bits[127:64])
11	000	0101	0001	001	AFSR1_EL1	Auxiliary Fault Status Register 1 (EL1)
11	100	0101	0001	001	AFSR1_EL2	Auxiliary Fault Status Register 1 (EL2)
11	110	0101	0001	001	AFSR1_EL3	Auxiliary Fault Status Register 1 (EL3)
11	000	0000	0010	001	ID_ISAR1_EL1	AArch32 Instruction Set Attribute Register
11	000	0010	0010	001	APDAKeyHi_EL1	Pointer Authentication Key A for Data (bits[127:64])
11	011	0100	0010	001	DAIF	Interrupt Mask Bits
11	011	1110	0010	001	CNTP_CTL_EL0	Counter-timer Physical Timer Control register
11	100	1110	0010	001	CNTHP_CTL_EL2	Counter-timer Hypervisor Physical Timer Control register
11	111	1110	0010	001	CNTPS_CTL_EL1	Counter-timer Physical Secure Timer Control register
11	000	0000	0011	001	MVFR1_EL1	AArch32 Media and VFP Feature Register
11	110	0001	0011	001	MDCR_EL3	Monitor Debug Configuration Register (EL3)
11	000	0010	0011	001	APGAKeyHi_EL1	Pointer Authentication Key A for Code (bits[127:64])
11	100	0100	0011	001	SPSR_abt	Saved Program Status Register (Abort mode)
11	011	1110	0011	001	CNTV_CTL_EL0	Counter-timer Virtual Timer Control register
11	100	1110	0011	001	CNTHV_CTL_EL2	Counter-timer Virtual Timer Control register (EL2)
11	000	0000	0100	001	ID_AA64PFR1_EL1	AArch64 Processor Feature Register 1
11	011	0100	0100	001	FPSR	Floating-point Status Register
11	000	1010	0100	001	LOREA_EL1	LORegion End Address (EL1)

op0	op1	Register selectors		op2	Name	Description
		CRn	CRm			
11	000	0000	0101	001	ID_AA64DFR1_EL1	AArch64 Debug Feature Register 1
11	011	0100	0101	001	DLR_EL0	Debug Link Register
11	000	0000	0110	001	ID_AA64ISAR1_EL1	AArch64 Instruction Set Attribute Register 1
11	000	0000	0111	001	ID_AA64MMFR1_EL1	AArch64 Memory Model Feature Register 1
11	000	1100	1000	001	ICC_EOIR0_EL1	Interrupt Controller Of Interrupt Register 0
11	000	1100	1000	001	ICV_EOIR0_EL1	Interrupt Controller Virtual End Of Interrupt Register 0
11	000	1001	1010	001	PMBPTR_EL1	Profiling Buffer Write Pointer Register
11	000	1100	1011	001	ICC_DIR_EL1	Interrupt Controller Deactivate Interrupt Register
11	000	1100	1011	001	ICV_DIR_EL1	Interrupt Controller Deactivate Virtual Interrupt Register
11	100	1100	1011	001	ICH_VTR_EL2	Interrupt Controller VGIC Type Register
11	011	1001	1100	001	PMCNTENSET_EL0	Performance Monitor Count Enable Set register
11	000	1100	1100	001	ICC_EOIR1_EL1	Interrupt Controller Of Interrupt Register 1
11	000	1100	1100	001	ICV_EOIR1_EL1	Interrupt Controller Virtual End Of Interrupt Register 1
11	011	1001	1101	001	PMXEVTYPER_EL0	Performance Monitor Selected Event Type Register
11	000	1001	1110	001	PMINTENSET_EL1	Performance Monitor Interrupt Enable Set register
10	000	0000	0000	010	OSDTRRX_EL1	OS Lock Data Transfer Register, Receive
11	001	0000	0000	010	CCSIDR2_EL1	Current Cache Size Register 2
11	000	0001	0000	010	CPACR_EL1	Architectural Feature Access Control Register
11	000	0010	0000	010	TCR_EL1	Translation Control Register (EL1)
11	100	0010	0000	010	TCR_EL2	Translation Control Register (EL2)
11	110	0010	0000	010	TCR_EL3	Translation Control Register (EL3)
11	000	1100	0000	010	RMR_EL1	Reset Management Register (EL1)
11	100	1100	0000	010	RMR_EL2	Reset Management Register (EL2)
11	110	1100	0000	010	RMR_EL3	Reset Management Register (EL3)
11	011	1101	0000	010	TPIDR_EL0	EL0 Read/Write Software Thread ID Register
11	100	1101	0000	010	TPIDR_EL2	EL2 Software Thread ID Register

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
11	110	1101	0000	010	TPIDR_EL3	EL3 Software Thread ID Register
11	011	1110	0000	010	CNTVCT_EL0	Counter-timer Virtual Timer Compare Value register
11	000	0000	0001	010	ID_DFR0_EL1	AArch32 Debug Feature Register 0
11	100	0001	0001	010	CPTR_EL2	Architectural Feature Trap Register (EL2)
11	110	0001	0001	010	CPTR_EL3	Architectural Feature Trap Register (EL3)
11	000	0010	0001	010	APIBKeyLo_EL1	Pointer Authentication Key B for Instructions (bits[63:0])
11	100	0010	0001	010	VTCCR_EL2	Virtualization Translation Control Register
10	000	0000	0010	010	MDSCR_EL1	Monitor Debug System Control Register
11	000	0000	0010	010	ID_ISAR2_EL1	AArch32 Instruction Set Attribute Register
11	000	0010	0010	010	APDBKeyLo_EL1	Pointer Authentication Key B for Data (bits[63:0])
11	000	0100	0010	010	CurrentEL	Current Exception Level
11	011	1110	0010	010	CNTP_CVAL_EL0	Counter-timer Physical Timer Compare Value register
11	100	1110	0010	010	CNTHP_CVAL_EL2	Counter-timer Hypervisor Physical Timer Compare Value register
11	111	1110	0010	010	CNTPS_CVAL_EL1	Counter-timer Physical Secure Timer Compare Value register
10	000	0000	0011	010	OSDTRTX_EL1	OS Lock Data Transfer Register, Transmit
11	000	0000	0011	010	MVFR2_EL1	AArch32 Media and VFP Feature Register
11	100	0100	0011	010	SPSR_und	Saved Program Status Register (Undefined mode)
11	011	1110	0011	010	CNTV_CVAL_EL0	Counter-timer Virtual Timer Compare Value register
11	100	1110	0011	010	CNTHV_CVAL_EL2	Counter-timer Virtual Timer Compare Value register (EL2)
11	000	1010	0100	010	LORN_EL1	LORegion Number (EL1)
10	000	0000	0110	010	OSECCR_EL1	OS Lock Exception Catch Control Register
11	000	0000	0111	010	ID_AA64MMFR2_EL1	AArch64 Memory Model Feature Register 2
11	000	1100	1000	010	ICC_HPPIR0_EL1	Interrupt Controller Highest Priority Pending Interrupt Register 0
11	000	1100	1000	010	ICV_HPPIR0_EL1	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0

op0	op1	Register selectors CRn	CRm	op2	Name	Description
						Pending Interrupt Register 0
11	000	1001	1001	010	PMSICR_EL1	Sampling Interval Counter Register
11	100	1100	1011	010	ICH_MISR_EL2	Interrupt Controller Maintenance Interrupt State Register
11	011	1001	1100	010	PMCNTENCLR_EL0	Performance Monitor Count Enable Clear register
11	000	1100	1100	010	ICC_HPIR1_EL1	Interrupt Controller Highest Priority Pending Interrupt Register 1
11	000	1100	1100	010	ICV_HPIR1_EL1	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
11	011	1001	1101	010	PMXEVCNTR_EL0	Performance Monitor Selected Event Counter Register
11	000	1001	1110	010	PMINTENCLR_EL1	Performance Monitor Interrupt Enable Clear register
11	011	1101	0000	011	TPIDRRO_EL0	EL0 Read-Only Software Thread ID Register
11	100	1110	0000	011	CNTVOFF_EL2	Counter-timer Virtual Offset register
11	000	0000	0001	011	ID_AFR0_EL1	AArch32 Auxiliary Feature Register 0
11	100	0001	0001	011	HSTR_EL2	Hypervisor System Register
11	000	0010	0001	011	APIBKeyHi_EL1	Pointer Authentication Key B for Instructions (bits[127:64])
11	000	0000	0010	011	ID_ISAR3_EL1	AArch32 Instruction Set Attribute Register
11	000	0010	0010	011	APDBKeyHi_EL1	Pointer Authentication Key B for Data (bits[127:64])
11	000	0100	0010	011	PAN	Privileged Access Never
11	100	0100	0011	011	SPSR_fiq	Saved Program Status Register (FIQ mode)
11	000	1010	0100	011	LORC_EL1	LORegion Control (EL1)
11	000	1100	1000	011	ICC_BPR0_EL1	Interrupt Controller Binary Point Register
11	000	1100	1000	011	ICV_BPR0_EL1	Interrupt Controller Virtual Binary Point Register 0
11	000	1001	1010	011	PMBSR_EL1	Profiling Buffer Status syndrome Register
11	000	1100	1011	011	ICC_RPR_EL1	Interrupt Controller Running Priority Register
11	000	1100	1011	011	ICV_RPR_EL1	Interrupt Controller Virtual Running Priority Register

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
11	100	1100	1011	011	ICH_EISR_EL2	Interrupt Controller of Interrupt Status Register
11	011	1001	1100	011	PMOVSCLR_EL0	Performance Monitor Overflow Flag Status Clear Register
11	000	1100	1100	011	ICC_BPR1_EL1	Interrupt Controller Binary Point Register
11	000	1100	1100	011	ICV_BPR1_EL1	Interrupt Controller Virtual Binary Point Register 1
11	011	1001	1110	011	PMOVSSET_EL0	Performance Monitor Overflow Flag Status Set register
10	000	0001	0000	100	OSLAR_EL1	OS Lock Access Register
11	100	0110	0000	100	HPFAR_EL2	Hypervisor IPA Fault Address Register
11	000	1101	0000	100	TPIDR_EL1	EL1 Software Thread ID Register
11	000	0000	0001	100	ID_MMFR0_EL1	AArch32 Memory Model Feature Register 0
10	000	0001	0001	100	OSLSR_EL1	OS Lock Status Register
11	000	0000	0010	100	ID_ISAR4_EL1	AArch32 Instruction Set Attribute Register
11	000	0100	0010	100	UAO	User Access Override
10	000	0001	0011	100	OSDLR_EL1	OS Double Lock Register
10	000	0001	0100	100	DBGPRCR_EL1	Debug Power Control Register
11	000	0000	0101	100	ID_AA64AFR0_EL1	AArch64 Auxiliary Feature Register 0
11	000	1001	1001	100	PMSFCR_EL1	Sampling Filter Control Register
11	011	1001	1100	100	PMSWINC_EL0	Performance Monitor Software Increment register
11	000	1100	1100	100	ICC_CTLR_EL1	Interrupt Controller Control Register (EL1)
11	000	1100	1100	100	ICV_CTLR_EL1	Interrupt Controller Virtual Control Register
11	110	1100	1100	100	ICC_CTLR_EL3	Interrupt Controller Control Register (EL3)
10	000	0000	xxxx	100	DBGBVR<n>_EL1	Debug Breakpoint Value Registers
11	000	0000	0000	101	MPIDR_EL1	Multiprocessor Affinity Register
11	100	0000	0000	101	VMPIDR_EL2	Virtualization Multiprocessor ID Register
11	000	0000	0001	101	ID_MMFR1_EL1	AArch32 Memory Model Feature Register 1
11	000	0000	0010	101	ID_ISAR5_EL1	AArch32 Instruction Set Attribute Register
11	000	0000	0101	101	ID_AA64AFR1_EL1	AArch64 Auxiliary Feature Register 1
11	000	1001	1001	101	PMSEVFR_EL1	Sampling Event Filter Register

op0	op1	Register selectors			Name	Description
CRn	CRm	op2				
11	100	1100	1001	101	ICC_SRE_EL2	Interrupt Controller System Register Enable register (EL2)
11	000	1100	1011	101	ICC_SGI1R_EL1	Interrupt Controller Software Generated Interrupt Group 1 Register
11	100	1100	1011	101	ICH_ELRSR_EL2	Interrupt Controller Empty List Register Status Register
11	011	1001	1100	101	PMSELR_EL0	Performance Monitoring Event Counter Selection Register
11	000	1100	1100	101	ICC_SRE_EL1	Interrupt Controller System Register Enable register (EL1)
11	110	1100	1100	101	ICC_SRE_EL3	Interrupt Controller System Register Enable register (EL3)
10	000	0000	xxxx	101	DBGBCR<n>_EL1	Debug Breakpoint Control Registers
11	000	0000	0000	110	REVIDR_EL1	Revision ID Register
11	000	0000	0001	110	ID_MMFR2_EL1	AArch32 Memory Model Feature Register 2
11	000	0000	0010	110	ID_MMFR4_EL1	AArch32 Memory Model Feature Register 4
10	000	0111	1000	110	DBGCLAIMSET_EL1	Debug Claim Tag Set register
10	000	0111	1001	110	DBGCLAIMCLR_EL1	Debug Claim Tag Clear register
11	000	1001	1001	110	PMSLATFR_EL1	Sampling Latency Filter Register
11	000	1100	1011	110	ICC_ASGI1R_EL1	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
11	011	1001	1100	110	PMCEID0_EL0	Performance Monitoring Common Event Identification register
11	000	1100	1100	110	ICC_IGRPEN0_EL1	Interrupt Controller Interrupt Group 0 Enable register
11	000	1100	1100	110	ICV_IGRPEN0_EL1	Interrupt Controller Virtual Interrupt Group 0 Enable register
10	000	0111	1110	110	DBGAUTHSTATUS_EL1	Debug Authentication Status register
10	000	0000	xxxx	110	DBGWVR<n>_EL1	Debug Watchpoint Value Registers
11	001	0000	0000	111	AIDR_EL1	Auxiliary ID Register
11	011	0000	0000	111	DCZID_EL0	Data Cache Zero ID register
11	000	0000	0001	111	ID_MMFR3_EL1	AArch32 Memory Model Feature Register 3
11	100	0001	0001	111	HACR_EL2	Hypervisor Auxiliary Control Register
11	000	0000	0010	111	ID_ISAR6_EL1	AArch32 Instruction Set Attribute Register

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
11	000	1010	0100	111	LORID_EL1	LORegionID (EL1)
11	000	1001	1001	111	PMSIDR_EL1	Sampling Profiling Register
11	000	1001	1010	111	PMBIDR_EL1	Profiling Buffer ID Register
11	000	1100	1011	111	ICC_SGI0R_EL1	Interrupt Controller Software Generated Interrupt Group 0 Register
11	100	1100	1011	111	ICH_VMCR_EL2	Interrupt Controller Virtual Machine Control Register
11	011	1001	1100	111	PMCEID1_EL0	Performance Monitoring Common Event Identification registers
11	000	1100	1100	111	ICC_IGRPEN1_EL1	Interrupt Controller Interrupt Group 1 Enable register
11	000	1100	1100	111	ICV_IGRPEN1_EL1	Interrupt Controller Virtual Interrupt Group 1 Enable register
11	110	1100	1100	111	ICC_IGRPEN1_EL3	Interrupt Controller Interrupt Group 1 Enable register (EL3)
11	011	1110	1111	111	PMCCFILTR_EL0	Performance Monitoring Cycle Count Filter Register
10	000	0000	xxxx	111	DBGWCR<n>_EL1	Debug Watchpoint Control Registers
11	100	1100	1000	0xx	ICH_AP0R<n>_EL2	Interrupt Controller Hyp Active Priorities Group 0 Registers
11	000	1100	1001	0xx	ICC_APIR<n>_EL1	Interrupt Controller Active Priorities Group 1 Registers
11	000	1100	1001	0xx	ICV_APIR<n>_EL1	Interrupt Controller Virtual Active Priorities Group 1 Registers
11	100	1100	1001	0xx	ICH_APIR<n>_EL2	Interrupt Controller Hyp Active Priorities Group 1 Registers
11	000	1100	1000	1xx	ICC_AP0R<n>_EL1	Interrupt Controller Active Priorities Group 0 Registers
11	000	1100	1000	1xx	ICV_AP0R<n>_EL1	Interrupt Controller Virtual Active Priorities Group 0 Registers
11	011	1110	10xx	xxx	PMEVCNTR<n>_EL0	Performance Monitoring Event Count Registers
11	100	1100	110x	xxx	ICH_LR<n>_EL2	Interrupt Controller Registers
11	011	1110	11xx	xxx	PMEVTYPER<n>_EL0	Performance Monitoring Event Type Registers
11	xxx	xxxx	xxxx	xxx	S3_<op1>_<Cn>_<Cm>_<op2>	IMPLEMENTATION DEFINED registers

Accessed using **SYS/SYSLAT**:

Register selectors		CRm		Name		Description
CRn	op1	op2	CRm	S1_<op1>_<Cn>_<Cm>_<op2>		IMPLEMENTATION DEFINED maintenance instructions
xxxx	xxx	xxx	xxxx			

Register selectors		CRm		Name		Description
op0	op1	CRn	CRm	op2		
01	000	0111	1000	000	AT S1E1R	Address Translate Stage 1 EL1 Read
01	100	0111	1000	000	AT S1E2R	Address Translate Stage 1 EL2 Read
01	110	0111	1000	000	AT S1E3R	Address Translate Stage 1 EL3 Read
01	000	0111	1000	001	AT S1E1W	Address Translate Stage 1 EL1 Write
01	100	0111	1000	001	AT S1E2W	Address Translate Stage 1 EL2 Write
01	110	0111	1000	001	AT S1E3W	Address Translate Stage 1 EL3 Write
01	000	0111	1000	010	AT S1E0R	Address Translate Stage 1 EL0 Read
01	000	0111	1000	011	AT S1E0W	Address Translate Stage 1 EL0 Write
01	100	0111	1000	100	AT S12E1R	Address Translate Stages 1 and 2 EL1 Read
01	100	0111	1000	101	AT S12E1W	Address Translate Stages 1 and 2 EL1 Write
01	100	0111	1000	110	AT S12E0R	Address Translate Stages 1 and 2 EL0 Read
01	100	0111	1000	111	AT S12E0W	Address Translate Stages 1 and 2 EL0 Write
01	000	0111	1001	000	AT S1E1RP	Address Translate Stage 1 EL1 Read PAN
01	000	0111	1001	001	AT S1E1WP	Address Translate Stage 1 EL1 Write PAN

Accessed using **TLBI**:

Register selectors		CRm		Name		Description
op0	op1	CRn	CRm	op2	Rt	
01	100	1000	0000	001	—	TLBI IPAS2E1IS
01	000	1000	0011	001	—	TLBI VAE1IS
01	100	1000	0011	001	—	TLBI VAE2IS
01	110	1000	0011	001	—	TLBI VAE3IS
01	100	1000	0100	001	—	TLBI IPAS2E1
01	000	1000	0111	001	—	TLBI VAE1
01	100	1000	0111	001	—	TLBI VAE2
01	110	1000	0111	001	—	TLBI VAE3
01	000	1000	0011	010	—	TLBI ASIDE1IS
01	000	1000	0111	010	—	TLBI ASIDE1
01	000	1000	0011	011	—	TLBI VAAE1IS
01	000	1000	0111	011	—	TLBI VAAE1
01	100	1000	0000	101	—	TLBI IPAS2LE1IS
01	000	1000	0011	101	—	TLBI VALE1IS
01	100	1000	0011	101	—	TLBI VALE2IS
01	110	1000	0011	101	—	TLBI VALE3IS
01	100	1000	0100	101	—	TLBI IPAS2LE1

Register selectors						Name	Description
op0	op1	CRn	CRm	op2	Rt		
01	000	1000	0111	101	—	TLBI VALE1	TLB Invalidate by VA, Last level, EL1
01	100	1000	0111	101	—	TLBI VALE2	TLB Invalidate by VA, Last level, EL2
01	110	1000	0111	101	—	TLBI VALE3	TLB Invalidate by VA, Last level, EL3
01	000	1000	0011	111	—	TLBI VAALE1IS	TLB Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable
01	000	1000	0111	111	—	TLBI VAALE1	TLB Invalidate by VA, All ASID, Last level, EL1
01	000	1000	0011	000	11111	TLBI VMALLE1IS	TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable
01	100	1000	0011	000	11111	TLBI ALLE2IS	TLB Invalidate All, EL2, Inner Shareable
01	110	1000	0011	000	11111	TLBI ALLE3IS	TLB Invalidate All, EL3, Inner Shareable
01	000	1000	0111	000	11111	TLBI VMALLE1	TLB Invalidate by VMID, All at stage 1, EL1
01	100	1000	0111	000	11111	TLBI ALLE2	TLB Invalidate All, EL2
01	110	1000	0111	000	11111	TLBI ALLE3	TLB Invalidate All, EL3
01	100	1000	0011	100	11111	TLBI ALLE1IS	TLB Invalidate All, EL1, Inner Shareable
01	100	1000	0111	100	11111	TLBI ALLE1	TLB Invalidate All, EL1
01	100	1000	0011	110	11111	TLBI VMALLS12E1IS	TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner Shareable
01	100	1000	0111	110	11111	TLBI VMALLS12E1	TLB Invalidate by VMID, All at Stage 1 and 2, EL1

28/0907/2017 0816:4140

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg v83A xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)[\(old\)](#)

htmldiff from-

[\(new\)](#)

SysReg_v83A_xml-00bet4

[SysReg_v83A_xml-00bet5](#)

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](#)
- [GIC](#)
- [GICD](#)
- [GICR](#)
- [GICC](#)
- [GICV](#)
- [GICH](#)
- [GITS](#)
- [Ptr Auth](#)

In the ID functional group:

Exec state	Name	Description
AArch32	AIDR	Auxiliary ID Register
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	FPSID	Floating-Point System ID register
AArch32	ID_AFR0	Auxiliary Feature Register 0
AArch32	ID_DFR0	Debug Feature Register 0
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
AArch32	ID_PFR0	Processor Feature Register 0
AArch32	ID_PFR1	Processor Feature Register 1
AArch32	MIDR	Main ID Register

Exec state	Name	Description
AArch32	MPIDR	Multiprocessor Affinity 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
AArch32	REVIDR	Revision ID Register
AArch32	TCMTR	TCM Type Register
AArch32	TLBTR	TLB Type Register
AArch32	VMPIDR	Virtualization Multiprocessor ID Register
AArch32	VPIDR	Virtualization Processor ID Register
AArch64	AIDR_EL1	Auxiliary ID 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	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_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_AFR0_EL1	AArch32 Auxiliary Feature Register 0
AArch64	ID_DFR0_EL1	AArch32 Debug Feature Register 0
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_PFR0_EL1	AArch32 Processor Feature Register 0
AArch64	ID_PFR1_EL1	AArch32 Processor Feature Register 1
AArch64	MIDR_EL1	Main ID Register
AArch64	MPIDR_EL1	Multiprocessor Affinity 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
AArch64	REVIDR_EL1	Revision ID Register
AArch64	VMPIDR_EL2	Virtualization Multiprocessor ID Register
AArch64	VPIDR_EL2	Virtualization Processor ID Register
External	EDAA32PFR	External Debug AArch32 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	AMAIR0	Auxiliary Memory Attribute Indirection Register 0
AArch32	AMAIR1	Auxiliary Memory Attribute Indirection Register 1
AArch32	CONTEXTIDR	Context ID Register
AArch32	DACR	Domain Access Control Register

Exec state	Name	Description
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	VTCT	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	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	VTCT_EL2	Virtualization Translation Control Register
AArch64	VTTBR_EL2	Virtualization Translation Table Base Register

In the Other functional group:

Exec state	Name	Description
AArch32	ACTLR	Auxiliary Control Register
AArch32	ACTLR2	Auxiliary Control Register 2
AArch32	CPACR	Architectural Feature Access Control Register
AArch32	HACTLR	Hyp Auxiliary Control Register
AArch32	HACTLR2	Hyp Auxiliary Control Register 2
AArch32	HSCTLR	Hyp System Control Register
AArch32	SCTLR	System Control Register
AArch64	ACTLR_EL1	Auxiliary Control Register (EL1)
AArch64	ACTLR_EL2	Auxiliary Control Register (EL2)
AArch64	ACTLR_EL3	Auxiliary Control Register (EL3)
AArch64	CPACR_EL1	Architectural Feature Access Control Register
AArch64	SCTLR_EL1	System Control Register (EL1)
AArch64	SCTLR_EL2	System Control Register (EL2)
AArch64	SCTLR_EL3	System 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	ESR_ELx	Exception Syndrome Register (ELx)
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	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	FPSCR	Floating-Point Status and Control Register
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	DLR_EL0	Debug Link Register
AArch64	DSPSR_EL0	Debug Saved Program Status Register
AArch64	ELR_EL1	Exception Link Register (EL1)
AArch64	ELR_EL2	Exception Link Register (EL2)
AArch64	ELR_EL3	Exception Link Register (EL3)
AArch64	FPCR	Floating-point Control Register
AArch64	FPSR	Floating-point Status Register
AArch64	SPSR_EL1	Saved Program Status Register (EL1)
AArch64	SPSR_EL2	Saved Program Status Register (EL2)
AArch64	SPSR_EL3	Saved Program Status Register (EL3)

Exec state	Name	Description
AArch64	SPSR_abt	Saved Program Status Register (Abort mode)
AArch64	SPSR_fiq	Saved Program Status Register (FIQ mode)
AArch64	SPSR_irq	Saved Program Status Register (IRQ mode)
AArch64	SPSR_und	Saved Program Status Register (Undefined mode)
AArch64	SP_EL0	Stack Pointer (EL0)
AArch64	SP_EL1	Stack Pointer (EL1)
AArch64	SP_EL2	Stack Pointer (EL2)
AArch64	SP_EL3	Stack Pointer (EL3)
AArch64	UAO	User Access Override

In the PSTATE functional group:

Exec state	Name	Description
AArch32	APSR	Application Program Status Register
AArch32	CPSR	Current Program Status Register
AArch64	CurrentEL	Current Exception Level
AArch64	DAIF	Interrupt Mask Bits
AArch64	NZCV	Condition Flags
AArch64	PAN	Privileged Access Never
AArch64	SPSel	Stack Pointer Select

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 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 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 ISW IC IALLU	Data or unified Cache line Invalidate by All Set/Way to PoU
AArch64	DC IVAC IC IALLUIS	Data or unified Cache line Invalidate by All VA to PoC PoU, Inner Shareable
AArch64	DC ZVA DC ISW	Data or unified Cache Zero line Invalidate by VA Set/Way
AArch64	IC IALLU DC IVAC	Instruction Data or unified Cache line Invalidate All by VA to PoU PoC
AArch64	IC IALLUIS IC IVAU	Instruction Cache line Invalidate All by VA to PoU, Inner Shareable
AArch64	IC IVAU DC ZVA	Instruction Data Cache line Zero 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

Exec state	Name	Description
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 S1E0R PAR_EL1	Physical Address Translate Stages 1 and 2 EL0 Read Register
AArch64	AT S1E0W AT S1E0R	Address Translate Stages 1 and 2 EL0 WriteRead
AArch64	AT S1E1R AT S1E0W	Address Translate Stages 1 and 2 EL1EL0 ReadWrite
AArch64	AT S1E1W AT S1E1R	Address Translate Stages 1 and 2 EL1 WriteRead
AArch64	AT S1E0R AT S1E1W	Address Translate StageStages 1 EL0and Read2 EL1 Write
AArch64	AT S1E0W AT S1E0R	Address Translate Stage 1 EL0 WriteRead
AArch64	AT S1E1R AT S1E0W	Address Translate Stage 1 EL1EL0 ReadWrite
AArch64	AT S1E1RP AT S1E1R	Address Translate Stage 1 EL1 Read PAN
AArch64	AT S1E1W AT S1E1RP	Address Translate Stage 1 EL1 WriteRead PAN
AArch64	AT S1E1WP AT S1E1W	Address Translate Stage 1 EL1 Write PAN
AArch64	AT S1E2R AT S1E1WP	Address Translate Stage 1 EL2EL1 ReadWrite PAN
AArch64	AT S1E2W AT S1E2R	Address Translate Stage 1 EL2 WriteRead
AArch64	AT S1E3R AT S1E2W	Address Translate Stage 1 EL3EL2 ReadWrite
AArch64	AT S1E3W AT S1E3R	Address Translate Stage 1 EL3 WriteRead
AArch64	PAR_EL1 AT S1E3W	Physical Address Register Translate Stage 1 EL3 Write

In the TLB functional group:

Exec state	Name	Description
AArch32	DTLBIALL	Data TLB Invalidate All
AArch32	DTLBIASID	Data TLB Invalidate by ASID match
AArch32	DTLBIMVA	Data TLB Invalidate by VA
AArch32	ITLBIALL	Instruction TLB Invalidate All
AArch32	ITLBIASID	Instruction TLB Invalidate by ASID match
AArch32	ITLBIMVA	Instruction TLB Invalidate by VA
AArch32	TLBIALL	TLB Invalidate All
AArch32	TLBIALLH	TLB Invalidate All, Hyp mode
AArch32	TLBIALLHIS	TLB Invalidate All, Hyp mode, Inner Shareable
AArch32	TLBIALLIS	TLB Invalidate All, Inner Shareable
AArch32	TLBIALLNSNH	TLB Invalidate All, Non-Secure Non-Hyp
AArch32	TLBIALLNSNHIS	TLB Invalidate All, Non-Secure Non-Hyp, Inner Shareable
AArch32	TLBIASID	TLB Invalidate by ASID match
AArch32	TLBIASIDIS	TLB Invalidate by ASID match, Inner Shareable
AArch32	TLBIIPAS2	TLB Invalidate by Intermediate Physical Address, Stage 2
AArch32	TLBIIPAS2IS	TLB Invalidate by Intermediate Physical Address, Stage 2, Inner Shareable
AArch32	TLBIIPAS2L	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level
AArch32	TLBIIPAS2LIS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, Inner Shareable
AArch32	TLBIMVA	TLB Invalidate by VA
AArch32	TLBIMVAA	TLB Invalidate by VA, All ASID
AArch32	TLBIMVAAIS	TLB Invalidate by VA, All ASID, Inner Shareable
AArch32	TLBIMVAAL	TLB Invalidate by VA, All ASID, Last level
AArch32	TLBIMVAALIS	TLB Invalidate by VA, All ASID, Last level, Inner Shareable
AArch32	TLBIMVAH	TLB Invalidate by VA, Hyp mode
AArch32	TLBIMVAHIS	TLB Invalidate by VA, Hyp mode, Inner Shareable
AArch32	TLBIMVAIS	TLB Invalidate by VA, Inner Shareable
AArch32	TLBIMVAL	TLB Invalidate by VA, Last level
AArch32	TLBIMVALH	TLB Invalidate by VA, Last level, Hyp mode
AArch32	TLBIMVALHIS	TLB Invalidate by VA, Last level, Hyp mode, Inner Shareable
AArch32	TLBIMVALIS	TLB Invalidate by VA, Last level, Inner Shareable
AArch64	TLBI ALLE1	TLB Invalidate All, EL1
AArch64	TLBI ALLE1IS	TLB Invalidate All, EL1, Inner Shareable
AArch64	TLBI ALLE2	TLB Invalidate All, EL2
AArch64	TLBI ALLE2IS	TLB Invalidate All, EL2, Inner Shareable
AArch64	TLBI ALLE3	TLB Invalidate All, EL3
AArch64	TLBI ALLE3IS	TLB Invalidate All, EL3, Inner Shareable
AArch64	TLBI ASIDE1	TLB Invalidate by ASID, EL1
AArch64	TLBI ASIDE1IS	TLB Invalidate by ASID, EL1, Inner Shareable
AArch64	TLBI IPAS2E1	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1

Exec state	Name	Description
AArch64	TLBI IPAS2E1IS	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
AArch64	TLBI IPAS2LE1	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
AArch64	TLBI IPAS2LE1IS	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
AArch64	TLBI VAAE1	TLB Invalidate by VA, All ASID, EL1
AArch64	TLBI VAAE1IS	TLB Invalidate by VA, All ASID, EL1, Inner Shareable
AArch64	TLBI VAALE1	TLB Invalidate by VA, All ASID, Last level, EL1
AArch64	TLBI VAALE1IS	TLB Invalidate by VA, All ASID, Last Level, EL1, Inner Shareable
AArch64	TLBI VAE1	TLB Invalidate by VA, EL1
AArch64	TLBI VAE1IS	TLB Invalidate by VA, EL1, Inner Shareable
AArch64	TLBI VAE2	TLB Invalidate by VA, EL2
AArch64	TLBI VAE2IS	TLB Invalidate by VA, EL2, Inner Shareable
AArch64	TLBI VAE3	TLB Invalidate by VA, EL3
AArch64	TLBI VAE3IS	TLB Invalidate by VA, EL3, Inner Shareable
AArch64	TLBI VALE1	TLB Invalidate by VA, Last level, EL1
AArch64	TLBI VALE1IS	TLB Invalidate by VA, Last level, EL1, Inner Shareable
AArch64	TLBI VALE2	TLB Invalidate by VA, Last level, EL2
AArch64	TLBI VALE2IS	TLB Invalidate by VA, Last level, EL2, Inner Shareable
AArch64	TLBI VALE3	TLB Invalidate by VA, Last level, EL3
AArch64	TLBI VALE3IS	TLB Invalidate by VA, Last level, EL3, Inner Shareable
AArch64	TLBI VMALLE1	TLB Invalidate by VMID, All at stage 1, EL1
AArch64	TLBI VMALLE1IS	TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable
AArch64	TLBI VMALLS12E1	TLB Invalidate by VMID, All at Stage 1 and 2, EL1
AArch64	TLBI VMALLS12E1IS	TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner 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	PMEVTYPER<n>	Performance Monitors Event Type Registers
AArch32	PMINTENCLR	Performance Monitors Interrupt Enable Clear register
AArch32	PMINTENSET	Performance Monitors Interrupt Enable Set register
AArch32	PMOVS	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	PMXVCNTR	Performance Monitors Selected Event Count Register
AArch32	PMXVETYP	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	PMEVTYPER<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
AArch64	PMOVSCLEL_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

Exec state	Name	Description
AArch64	PMXEVCNTR_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	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	PMOVSCLR_EL0	Performance Monitors Overflow Flag Status Clear register
External	PMOVSSET_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

Exec state	Name	Description
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	AMAIR0	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	HAMAIR0	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	S1_<op1>_<Cn>_<Cm>_<op2>	IMPLEMENTATION DEFINED maintenance instructions
AArch64	S3_<op1>_<Cn>_<Cm>_<op2>	IMPLEMENTATION DEFINED registers

In the Timer functional group:

Exec state	Name	Description
AArch32	CNTFRQ	Counter-timer Frequency register
AArch32	CNTHCTL	Counter-timer Hyp Control register
AArch32	CNTHP_CTL	Counter-timer Hyp Physical Timer Control register
AArch32	CNTHP_CVAL	Counter-timer Hyp Physical CompareValue register
AArch32	CNTHP_TVAL	Counter-timer Hyp Physical Timer TimerValue register
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	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	CNTVOFF	Counter-timer Virtual Offset 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	CNTHCTL_EL2	Counter-timer Hypervisor Control register

Exec state	Name	Description
AArch64	CNTHP_CTL_EL2	Counter-timer Hypervisor Physical Timer Control register
AArch64	CNTHP_CVAL_EL2	Counter-timer Hypervisor Physical Timer CompareValue register
AArch64	CNTHP_TVAL_EL2	Counter-timer Hypervisor Physical Timer TimerValue register
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	CNTPCT_EL0	Counter-timer Physical Count 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	CNTVCT_EL0	Counter-timer Virtual Count register
AArch64	CNTVOFF_EL2	Counter-timer Virtual Offset 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	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	CNTSR	Counter Status Register
External	CNTTIDR	Counter-timer Timer ID Register
External	CNTVCT	Counter-timer Virtual Count
External	CNTVOFF	Counter-timer Virtual Offset
External	CNTVOFF<n>	Counter-timer Virtual Offsets
External	CNTV_CTL	Counter-timer Virtual Timer Control
External	CNTV_CVAL	Counter-timer Virtual Timer CompareValue
External	CNTV_TVAL	Counter-timer Virtual Timer TimerValue
External	CounterID<n>	Counter ID registers

In the Debug functional group:

Exec state	Name	Description
AArch32	DBGAUTHSTATUS	Debug Authentication Status register
AArch32	DBGBCR<n>	Debug Breakpoint Control Registers
AArch32	DBGBVR<n>	Debug Breakpoint Value Registers
AArch32	DBGBXVR<n>	Debug Breakpoint Extended Value Registers
AArch32	DBGCLAIMCLR	Debug Claim Tag Clear register
AArch32	DBGCLAIMSET	Debug Claim Tag Set register
AArch32	DBGDCCINT	DCC Interrupt Enable Register
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

Exec state	Name	Description
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	DLR	Debug Link Register
AArch32	DSPSR	Debug Saved Program Status Register
AArch32	HDCR	Hyp Debug Control Register
AArch32	SDCR	Secure Debug Control Register
AArch32	SDER	Secure Debug Enable 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	MDCR_EL2	Monitor Debug Configuration Register (EL2)
AArch64	MDCR_EL3	Monitor Debug Configuration Register (EL3)
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	SDER32_EL3	AArch32 Secure Debug Enable Register
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

Exec state	Name	Description
External	EDECR	External Debug Execution Control Register
External	EDESR	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	CTIAPPULSE	CTI Application Pulse register
External	CTIAPPSET	CTI Application Trigger Set register
External	CTIAUTHSTATUS	CTI Authentication Status register
External	CTICHINSTATUS	CTI Channel In Status register
External	CTICHOUTSTATUS	CTI Channel Out Status register
External	CTICIDR0	CTI Component Identification Register 0
External	CTICIDR1	CTI Component Identification Register 1
External	CTICIDR2	CTI Component Identification Register 2
External	CTICIDR3	CTI Component Identification Register 3
External	CTICLAIMCLR	CTI Claim Tag Clear register
External	CTICLAIMSET	CTI Claim Tag Set register
External	CTICONTROL	CTI Control register
External	CTIDEVAFF0	CTI Device Affinity register 0
External	CTIDEVAFF1	CTI Device Affinity register 1
External	CTIDEVARCH	CTI Device Architecture register
External	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	HTTBR	Hyp Translation Table Base Register
AArch32	HVBAR	Hyp Vector Base Address Register
AArch32	ICC_HSRE	Interrupt Controller Hyp System Register Enable register
AArch32	ICH_AP0R<n>	Interrupt Controller Hyp Active Priorities Group 0 Registers
AArch32	ICH_AP1R<n>	Interrupt Controller Hyp Active Priorities Group 1 Registers
AArch32	ICH_EISR	Interrupt Controller End of Interrupt Status Register
AArch32	ICH_ELRSR	Interrupt Controller Empty List Register Status Register
AArch32	ICH_HCR	Interrupt Controller Hyp Control Register
AArch32	ICH_LR<n>	Interrupt Controller List Registers
AArch32	ICH_LRC<n>	Interrupt Controller List Registers
AArch32	ICH_MISR	Interrupt Controller Maintenance Interrupt State Register
AArch32	ICH_VMCR	Interrupt Controller Virtual Machine Control Register
AArch32	ICH_VTR	Interrupt Controller VGIC Type Register
AArch32	TLBIALLH	TLB Invalidate All, Hyp mode
AArch32	TLBIALLHIS	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	VTCTCR	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	CNTHP_CTL_EL2	Counter-timer Hypervisor Physical Timer Control register

Exec state	Name	Description
AArch64	CNTHP_CVAL_EL2	Counter-timer Hypervisor Physical Timer CompareValue register
AArch64	CNTHP_TVAL_EL2	Counter-timer Hypervisor Physical Timer TimerValue register
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	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 TLBI IPAS2EI	Memory TLB Attribute Invalidate Indirection by Register Intermediate (EL2) Physical Address, Stage 2, EL1
AArch64	MDCR_EL2 TLBI IPAS2EHS	Monitor TLB Debug Invalidate Configuration by Register Intermediate (EL2) Physical Address, Stage 2, EL1, Inner Shareable
AArch64	RMR_EL2 TLBI IPAS2LEI	Reset TLB Management Invalidate Register by (EL2) Intermediate Physical Address, Stage 2, Last level, EL1
AArch64	SCTLR_EL2 TLBI IPAS2LEHS	System TLB Control Invalidate Register by (EL2) Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
AArch64	TCR_EL2 MAIR_EL2	Translation Memory Control Attribute Indirection Register (EL2)
AArch64	TLBI IPAS2EI MDCR_EL2	TLB Monitor Invalidate Debug by Configuration Intermediate Register Physical Address, Stage 2, EL1(EL2)
AArch64	TLBI IPAS2EIIS RMR_EL2	TLB Reset Invalidate Management by Register Intermediate Physical Address, Stage 2, EL1, Inner Shareable(EL2)
AArch64	TLBI IPAS2LEI SCTLR_EL2	TLB System Invalidate Control by Register Intermediate Physical Address, Stage 2, Last level, EL1(EL2)
AArch64	TLBI IPAS2LEIIS TCR_EL2	TLB Translation Invalidate Control by Register Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable(EL2)
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)

Exec state	Name	Description
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 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_HPIR0	Interrupt Controller Highest Priority Pending Interrupt Register 0
AArch32	ICC_HPIR1	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

Exec state	Name	Description
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_APIR<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_HPIR0	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
AArch32	ICV_HPIR1	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_APIR<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_HPIR0_EL1	Interrupt Controller Highest Priority Pending Interrupt Register 0
AArch64	ICC_HPIR1_EL1	Interrupt Controller Highest Priority Pending Interrupt Register 1
AArch64	ICC_IAR0_EL1	Interrupt Controller Interrupt Acknowledge Register 0
AArch64	ICC_IAR1_EL1	Interrupt Controller Interrupt Acknowledge Register 1
AArch64	ICC_IGRPEN0_EL1	Interrupt Controller Interrupt Group 0 Enable register
AArch64	ICC_IGRPEN1_EL1	Interrupt Controller Interrupt Group 1 Enable register
AArch64	ICC_IGRPEN1_EL3	Interrupt Controller Interrupt Group 1 Enable register (EL3)
AArch64	ICC_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_APIR<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_APIR<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

Exec state	Name	Description
AArch64	ICV_EOIR1_EL1	Interrupt Controller Virtual End Of Interrupt Register 1
AArch64	ICV_HPIR0_EL1	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
AArch64	ICV_HPIR1_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_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_ICENABLER<n>	Interrupt Clear-Enable Registers
External	GICD_ICFGR<n>	Interrupt Configuration Registers
External	GICD_ICPENDR<n>	Interrupt Clear-Pending Registers
External	GICD_IGROUPR<n>	Interrupt Group Registers
External	GICD_IGRPMODR<n>	Interrupt Group Modifier Registers
External	GICD_IIDR	Distributor Implementer Identification Register
External	GICD_IPRIORITYR<n>	Interrupt Priority Registers
External	GICD_IROUTER<n>	Interrupt Routing Registers
External	GICD_ISACTIVER<n>	Interrupt Set-Active Registers
External	GICD_ISENABLER<n>	Interrupt Set-Enable Registers
External	GICD_ISPENDR<n>	Interrupt Set-Pending Registers
External	GICD_ITARGETSR<n>	Interrupt Processor Targets Registers
External	GICD_NSACR<n>	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>	SPI Set-Pending Registers
External	GICD_STATUSR	Error Reporting Status Register
External	GICD_TYPER	Interrupt Controller 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_ICENABLER0	Interrupt Clear-Enable Register 0
External	GICR_ICFGR0	Interrupt Configuration Register 0
External	GICR_ICFGR1	Interrupt Configuration Register 1
External	GICR_ICPENDR0	Interrupt Clear-Pending Register 0
External	GICR_IGROUPR0	Interrupt Group Register 0
External	GICR_IGRPMODR0	Interrupt Group Modifier Register 0
External	GICR_IIDR	Redistributor Implementer Identification Register
External	GICR_INVALIDLR	Redistributor Invalidate All Register
External	GICR_INVALIDPIR	Redistributor Invalidate LPI Register
External	GICR_IPRIORITYR<n>	Interrupt Priority Registers
External	GICR_ISACTIVER0	Interrupt Set-Active Register 0
External	GICR_ISENABLER0	Interrupt Set-Enable Register 0
External	GICR_ISPENDR0	Interrupt Set-Pending Register 0
External	GICR_NSACR	Non-secure Access Control Register
External	GICR_PENDBASER	Redistributor LPI Pending Table Base Address Register
External	GICR_PROPBASER	Redistributor Properties Base Address Register
External	GICR_SETLPIR	Set LPI Pending Register
External	GICR_STATUSR	Error Reporting Status Register

Exec state	Name	Description
External	GICR_SYNCRR	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_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_HPPPIR	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_HPPPIR	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_TRANSLATER	ITS Translation Register
External	GITS_TYPER	ITS Type Register

In the Ptr Auth functional group:

Exec state	Name	Description
AArch64	APDAKeyHi_EL1	Pointer Authentication Key A for Data (bits[127:64])
AArch64	APDAKeyLo_EL1	Pointer Authentication Key A for Data (bits[63:0])
AArch64	APDBKeyHi_EL1	Pointer Authentication Key B for Data (bits[127:64])
AArch64	APDBKeyLo_EL1	Pointer Authentication Key B for Data (bits[63:0])
AArch64	APGAKeyHi_EL1	Pointer Authentication Key A for Code (bits[127:64])
AArch64	APGAKeyLo_EL1	Pointer Authentication Key A for Code (bits[63:0])
AArch64	APIAKeyHi_EL1	Pointer Authentication Key A for Instruction (bits[127:64])
AArch64	APIAKeyLo_EL1	Pointer Authentication Key A for Instruction (bits[63:0])
AArch64	APIBKeyHi_EL1	Pointer Authentication Key B for Instruction (bits[127:64])
AArch64	APIBKeyLo_EL1	Pointer Authentication Key B for Instruction (bits[63:0])

28/0907/2017 0816:4140

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg v83A xml-00bet5](#)

[SysReg v83A xml-00bet4](#)[\(old\)](#)htmldiff from-
SysReg_v83A_xml-00bet4[\(new\)](#)[SysReg v83A xml-00bet5](#)

External System registers

ASICCTL: CTI External Multiplexer Control register

CNTACR<n>: Counter-timer Access Control Registers

CNTCR: Counter Control Register

CNTCV: Counter Count Value register

CNTEL0ACR: Counter-timer EL0 Access Control Register

CNTFID0: Counter Frequency ID

[CNTFID<n>](#): Counter Frequency IDs, [n > 0](#)

CNTFRQ: Counter-timer Frequency

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

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

CTIAPPCLEAR: CTI Application Trigger Clear register

CTIAPPPULSE: 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

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

CounterID<n>: Counter ID registers

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 AArch32 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

EDESr: 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

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_HPIR: 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_ICENABLER<n>: Interrupt Clear-Enable Registers
 GICD_ICFGR<n>: Interrupt Configuration Registers
 GICD_ICPENDR<n>: Interrupt Clear-Pending Registers
 GICD_IGROUPR<n>: Interrupt Group Registers
 GICD_IGRPMODR<n>: Interrupt Group Modifier Registers
 GICD_IIDR: Distributor Implementer Identification Register
 GICD_IPRIORITYR<n>: Interrupt Priority Registers
 GICD_IROUTER<n>: Interrupt Routing Registers
 GICD_ISACTIVER<n>: Interrupt Set-Active Registers
 GICD_ISENABLER<n>: Interrupt Set-Enable Registers
 GICD_ISPENDR<n>: Interrupt Set-Pending Registers
 GICD_ITARGETSR<n>: Interrupt Processor Targets Registers
 GICD_NSACR<n>: 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

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
 GICR_CLRLPIR: Clear LPI Pending Register
 GICR_CTLR: Redistributor Control Register
 GICR_ICACTIVER0: Interrupt Clear-Active Register 0
 GICR_ICENABLER0: Interrupt Clear-Enable Register 0
 GICR_ICFGR0: Interrupt Configuration Register 0
 GICR_ICFGR1: Interrupt Configuration Register 1
 GICR_ICPENDR0: Interrupt Clear-Pending Register 0
 GICR_IGROUPR0: Interrupt Group Register 0
 GICR_IGRPMODR0: Interrupt Group Modifier Register 0
 GICR_IIDR: Redistributor Implementer Identification Register
 GICR_INVALLR: Redistributor Invalidate All Register
 GICR_INVLPIR: Redistributor Invalidate LPI Register
 GICR_IPRIORITYR<n>: Interrupt Priority Registers
 GICR_ISACTIVER0: Interrupt Set-Active Register 0
 GICR_ISENABLER0: Interrupt Set-Enable Register 0
 GICR_ISPENDR0: Interrupt Set-Pending Register 0
 GICR_NSACR: Non-secure Access Control 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_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_TRANSLATER: ITS Translation Register

GITS_TYPER: ITS Type Register

[MIDR_EL1](#): Main ID 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
 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
 PMOVSCLR_EL0: Performance Monitors Overflow Flag Status Clear register
 PMOVSSET_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

28/09/2017 08:46:41

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg v83A xml-00bet4](#)
[\(old\)](#)

htldiff from-
 SysReg_v83A_xml-00bet4

[\(new\)](#)
[SysReg v83A xml-00bet5](#)

SysReg_v83A_xml-00bet4

(old)

htmldiff from-

(new)

SysReg_v83A_xml-00bet4

SysReg_v83A_xml-00bet5

External register index by offset

Below are indexes for external registers in the following blocks:

- CTIPMU
- DebugGIC CPU interface
- GIC CPU interfaceGIC Virtual interface control
- GIC DistributorTimer
- GIC ITS controlDebug
- GIC ITS translationGIC Redistributor
- GIC RedistributorGIC Virtual CPU interface
- GIC Virtual CPU interfaceGIC ITS control
- GIC Virtual interface controlGIC ITS translation
- PMUCTI
- TimerGIC Distributor

In the CTIPMU block:

Offset	Name	Description
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
0x000 + 8n	CTICONTROL PMEVCNTR<n>_EL0	CTIPerformance ControlMonitors registerEvent Count Registers
0x0100x0F8	CTIINTACK PMCCNTR_EL0[31:0]	CTIPerformance OutputMonitors TriggerCycle Acknowledge registerCounter
0x0140x0FC	CTIAPPSET PMCCNTR_EL0[63:32]	CTIPerformance ApplicationMonitors TriggerCycle Set registerCounter
0x0180x200	CTIAPPCLEAR PMPCSR[31:0]	CTIProgram ApplicationCounter TriggerSample Clear registerRegister
0x01C0x204	CTIAPPULSE PMPCSR[63:32]	CTIProgram ApplicationCounter PulseSample registerRegister
0x020 + 4n0x208	CTIINEN<n> PMCID1SR	CTICONTEXTIDR_EL1 InputSample Trigger to Output Channel Enable registersRegister
0x0A0 + 4n0x20C	CTIOUTEN<n> PMVIDSR	CTIVMID InputSample Channel to Output Trigger Enable registersRegister
0x1300x220	CTITRIGINSTATUS PMPCSR[31:0]	CTIProgram TriggerCounter InSample Status registerRegister
0x1340x224	CTITRIGOUTSTATUS PMPCSR[63:32]	CTIProgram TriggerCounter OutSample Status registerRegister
0x1380x228	CTICHINSTATUS PMCID1SR	CTICONTEXTIDR_EL1 ChannelSample In Status registerRegister
0x13C0x22C	CTICHOUTSTATUS PMCID2SR	CTICONTEXTIDR_EL2 ChannelSample Out Status registerRegister
0x1400x400 + 4n	CTIGATE PMEVTPER<n>_EL0	CTIPerformance ChannelMonitors GateEvent EnableType registerRegisters
0x1440x47C	ASICCTL PMCCFILTR_EL0	CTIPerformance ExternalMonitors MultiplexerCycle ControlCounter registerFilter Register
0xF000xC00	CTIITCTRL PMCNTESET_EL0	CTIPerformance IntegrationMonitors modeCount ControlEnable-Set register

Offset	Name	Description
0xFA0-0xC20	CTICLAIMSET PMCNTENCLR_EL0	CTIPerformance ClaimMonitors TagCount SetEnable-Clear register
0xFA4-0xC40	CTICLAIMCLR PMINTENSET_EL1	CTIPerformance ClaimMonitors TagInterrupt ClearEnable-Set register
0xFA8-0xC60	CTIDEVAFF0 PMINTENCLR_EL1	CTIPerformance DeviceMonitors AffinityInterrupt Enable-Clear register 0
0xFAC-0xC80	CTIDEVAFF1 PMOVSLR_EL0	CTIPerformance DeviceMonitors AffinityOverflow Flag-Status-Clear register 1
0xFB0-0xCA0	CTILAR PMSWINC_EL0	CTIPerformance LockMonitors AccessSoftware Register-Increment register
0xFB4-0xCC0	CTILSR PMOVSSET_EL0	CTIPerformance LockMonitors Overflow-Flag-Status Register-Set register
0xFB8-0xE00	CTIAUTHSTATUS PMCFGR	CTIPerformance AuthenticationMonitors StatusConfiguration registerRegister
0xFBC-0xE04	CTIDEVARCH PMCR_EL0	CTIPerformance DeviceMonitors ArchitectureControl registerRegister
0xFC0-0xE20	CTIDEVID2 PMCEID0	CTIPerformance DeviceMonitors IDCommon Event Identification register 20
0xFC4-0xE24	CTIDEVID1 PMCEID1	CTIPerformance DeviceMonitors IDCommon Event Identification register 1
0xFC8-0xE28	CTIDEVID PMCEID2	CTIPerformance DeviceMonitors IDCommon Event Identification register 02
0xFCC-0xE2C	CTIDEVTYPE PMCEID3	CTIPerformance DeviceMonitors TypeCommon Event Identification register 3
0xFD0-0xF00	CTIPIDR4 PMITCTRL	CTIPerformance PeripheralMonitors IdentificationIntegration Registermode 4Control register
0xFE0-0xFA8	CTIPIDR0 PMDEVAFF0	CTIPerformance PeripheralMonitors IdentificationDevice RegisterAffinity register 0
0xFE4-0xFAC	CTIPIDR1 PMDEVAFF1	CTIPerformance PeripheralMonitors IdentificationDevice RegisterAffinity register 1
0xFE8-0xFB0	CTIPIDR2 PMLAR	CTIPerformance PeripheralMonitors IdentificationLock Access-Register 2
0xFEC-0xFB4	CTIPIDR3 PMLSR	CTIPerformance PeripheralMonitors IdentificationLock Status-Register 3
0xFF0-0xFB8	CTICIDR0 PMAUTHSTATUS	CTIPerformance ComponentMonitors IdentificationAuthentication RegisterStatus 0register
0xFF4-0xFBC	CTICIDR1 PMDEVARCH	CTIPerformance ComponentMonitors IdentificationDevice RegisterArchitecture 1register
0xFF8-0xFC8	CTICIDR2 PMDEVID	CTIPerformance ComponentMonitors IdentificationDevice RegisterID 2register
0xFFC-0xFCC	CTICIDR3 PMDEVTYPE	CTIPerformance ComponentMonitors IdentificationDevice RegisterType 3register

In the Debug block:

Offset	Name	Description
0x020	EDESR	External Debug Event Status Register
0x024	EDECR	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	EDCIDSr	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

Offset	Name	Description
0x310	EDPRCR	External Debug Power/Reset Control Register
0x314	EDPRSR	External Debug Processor Status Register
0x400 + 16n	DBGBVR<n>_EL1[31:0]	Debug Breakpoint Value Registers
0x404 + 16n	DBGBVR<n>_EL1[63:32]	Debug Breakpoint Value Registers
0x408 + 16n	DBGBCR<n>_EL1	Debug Breakpoint Control Registers
0x800 + 16n	DBGWVR<n>_EL1[31:0]	Debug Watchpoint Value Registers
0x804 + 16n	DBGWVR<n>_EL1[63:32]	Debug Watchpoint Value Registers
0x808 + 16n	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 AArch32 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 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_HPPIR	CPU Interface Highest Priority Pending Interrupt Register
0x001C	GICC_ABPR	CPU Interface Aliased Binary Point Register
0x0020-0x003C	GICC_AIAR	CPU Interface Aliased Interrupt Acknowledge Register

Offset	Name	Description
0x0024	GICC_AEOIR	CPU Interface Aliased End Of Interrupt Register
0x0028	GICC_AHPPIR	CPU Interface Aliased Highest Priority Pending Interrupt Register
0x002C	GICC_STATUSR	CPU Interface Status Register
0x00D0 + 4n	GICC_APR<n>	CPU Interface Active Priorities Registers
0x00E0 + 4n	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 GIC Distributor Virtual interface control block:

Offset	Name	Description
0x0000	GICD_CTLR GICH_HCR	Distributor Hypervisor Control Register
0x0004	GICD_TYPER GICH_VTR	Interrupt Controller Virtual Type Register
0x0008	GICD_IIDR GICH_VMCR	Distributor Virtual Implementer Machine Identification Control Register
0x0010	GICD_STATUSR GICH_MISR	Error Maintenance ReportingInterrupt Status Register
0x0040 0x0020	GICD_SETSPI_NSR GICH_EISR	Set End Non-secureInterrupt SPI PendingStatus Register
0x0048 0x0030	GICD_CLRSPI_NSR GICH_ELSR	Clear Empty Non-secureList SPIRegister PendingStatus Register
0x0050 0x00F0 + 4n	GICD_SETSPI_SR GICH_APR<n>	Set Active SecurePriorities SPI Pending RegisterRegisters
0x0058 0x0100 + 4n	GICD_CLRSPI_SR GICH_LR<n>	Clear List Secure SPI Pending RegisterRegisters
0x0080 + 4n	GICD_IGROUPR<n>	Interrupt Group Registers
0x0100 + 4n	GICD_ISENABLER<n>	Interrupt Set-Enable Registers
0x0180 + 4n	GICD_ICENABLER<n>	Interrupt Clear-Enable Registers
0x0200 + 4n	GICD_ISPENDR<n>	Interrupt Set-Pending Registers
0x0280 + 4n	GICD_ICPENDR<n>	Interrupt Clear-Pending Registers
0x0300 + 4n	GICD_ISACTIVER<n>	Interrupt Set-Active Registers
0x0380 + 4n	GICD_ICACTIVER<n>	Interrupt Clear-Active Registers
0x0400 + 4n	GICD_IPRIORITYR<n>	Interrupt Priority Registers
0x0800 + 4n	GICD_ITARGETSR<n>	Interrupt Processor Targets Registers
0x0C00 + 4n	GICD_ICFGR<n>	Interrupt Configuration Registers
0x0D00 + 4n	GICD_IGRPMODR<n>	Interrupt Group Modifier Registers
0x0E00 + 4n	GICD_NSACR<n>	Non-secure Access Control Registers
0x0F00	GICD_SGIR	Software Generated Interrupt Register
0x0F10 + 4n	GICD_CPENDSGIR<n>	SGI Clear-Pending Registers
0x0F20 + 4n	GICD_SPENDSGIR<n>	SGI Set-Pending Registers
0x6000 + 8n	GICD_IROUTER<n>	Interrupt Routing Registers

In the GIC ITS controlTimer block:

Offset	Name	Description
0x0000	GITS_CTLR	ITS Control Register
0x0004	GITS_IIDR	ITS Identification Register
0x0008-0x000C	GITS_TYPER	ITS Type Register
0x0080-0x0084	GITS_CBASER	ITS Command Queue Descriptor
0x0088-0x008C	GITS_CWRITER	ITS Write Register
0x0090-0x0094	GITS_CREADR	ITS Read Register
0x0100 + 8n	GITS_BASER<n>	ITS Translation Table Descriptors

Frame	Offset	Name	Description
CNTControlBase	0x000	CNTCR	Counter Control Register
CNTControlBase	0x004	CNTSR	Counter Status Register
CNTControlBase	0x008	CNTCV[31:0]	Counter Count Value register
CNTControlBase	0x00C	CNTCV[63:32]	Counter Count Value register
CNTControlBase	0x020	CNTFID0	Counter Frequency ID
CNTControlBase	0x020 + 4n	CNTFID<n>	Counter Frequency IDs
CNTControlBase	0xFD0 + 4n	CounterID<n>	Counter ID registers
CNTReadBase	0x000	CNTCV[31:0]	Counter Count Value register
CNTReadBase	0x004	CNTCV[63:32]	Counter Count Value register
CNTReadBase	0xFD0 + 4n	CounterID<n>	Counter ID registers
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 Compare Value
CNTBaseN	0x024	CNTP_CVAL[63:32]	Counter-timer Physical Timer Compare Value
CNTBaseN	0x028	CNTP_TVAL	Counter-timer Physical Timer Timer Value
CNTBaseN	0x02C	CNTP_CTL	Counter-timer Physical Timer Control
CNTBaseN	0x030	CNTV_CVAL[31:0]	Counter-timer Virtual Timer Compare Value
CNTBaseN	0x034	CNTV_CVAL[63:32]	Counter-timer Virtual Timer Compare Value
CNTBaseN	0x038	CNTV_TVAL	Counter-timer Virtual Timer Timer Value
CNTBaseN	0x03C	CNTV_CTL	Counter-timer Virtual Timer Control
CNTBaseN	0xFD0 + 4n	CounterID<n>	Counter ID registers
CNTEL0BaseN	0x000	CNTPCT[31:0]	Counter-timer Physical Count
CNTEL0BaseN	0x004	CNTPCT[63:32]	Counter-timer Physical Count
CNTEL0BaseN	0x008	CNTVCT[31:0]	Counter-timer Virtual Count
CNTEL0BaseN	0x00C	CNTVCT[63:32]	Counter-timer Virtual Count
CNTEL0BaseN	0x010	CNTFRQ	Counter-timer Frequency
CNTEL0BaseN	0x020	CNTP_CVAL[31:0]	Counter-timer Physical Timer Compare Value
CNTEL0BaseN	0x024	CNTP_CVAL[63:32]	Counter-timer Physical Timer Compare Value
CNTEL0BaseN	0x028	CNTP_TVAL	Counter-timer Physical Timer Timer Value
CNTEL0BaseN	0x02C	CNTP_CTL	Counter-timer Physical Timer Control
CNTEL0BaseN	0x030	CNTV_CVAL[31:0]	Counter-timer Virtual Timer Compare Value
CNTEL0BaseN	0x034	CNTV_CVAL[63:32]	Counter-timer Virtual Timer Compare Value
CNTEL0BaseN	0x038	CNTV_TVAL	Counter-timer Virtual Timer Timer Value
CNTEL0BaseN	0x03C	CNTV_CTL	Counter-timer Virtual Timer Control
CNTEL0BaseN	0xFD0 + 4n	CounterID<n>	Counter ID registers
CNTCTLBBase	0x000	CNTFRQ	Counter-timer Frequency
CNTCTLBBase	0x004	CNTNSAR	Counter-timer Non-secure Access Register
CNTCTLBBase	0x008	CNTTIDR	Counter-timer Timer ID Register
CNTCTLBBase	0x040 + 4n	CNTACR<n>	Counter-timer Access Control Registers
CNTCTLBBase	0x080 + 8n	CNTVOFF<n>[31:0]	Counter-timer Virtual Offsets
CNTCTLBBase	0x084 + 8n	CNTVOFF<n>[63:32]	Counter-timer Virtual Offsets
CNTCTLBBase	0xFD0 + 4n	CounterID<n>	Counter ID registers

In the **GIC ITS translation** **Debug** block:

Offset	Name	Description
0x024	EDECR	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 + 16n	DBGBVR<n>_EL1[31:0]	Debug Breakpoint Value Registers
0x404 + 16n	DBGBVR<n>_EL1[63:32]	Debug Breakpoint Value Registers
0x408 + 16n	DBGBCR<n>_EL1	Debug Breakpoint Control Registers
0x800 + 16n	DBGWVR<n>_EL1[31:0]	Debug Watchpoint Value Registers
0x804 + 16n	DBGWVR<n>_EL1[63:32]	Debug Watchpoint Value Registers
0x808 + 16n	DBGWCR<n>_EL1	Debug Watchpoint Control Registers
0xD00	MDR_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 AArch32 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

Offset	Name	Description
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
0x0040-0x020	GITS_TRANSLATER EDESR	ITSE External Translation Debug Event Status 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	0x0008-0x000C	GICR_TYPER	Redistributor Type Register
RD_base	0x0010	GICR_STATUSR	Error Reporting Status Register
RD_base	0x0014	GICR_WAKER	Redistributor Wake Register
RD_base	0x0040-0x0044	GICR_SETLPIR	Set LPI Pending Register
RD_base	0x0048-0x004C	GICR_CLRLPIR	Clear LPI Pending Register
RD_base	0x0070-0x0074	GICR_PROPBASER	Redistributor Properties Base Address Register
RD_base	0x0078-0x007C	GICR_PENDBASER	Redistributor LPI Pending Table Base Address Register
RD_base	0x00A0-0x00A4	GICR_INVLPIR	Redistributor Invalidate LPI Register
RD_base	0x00B0-0x00B4	GICR_INVALLR	Redistributor Invalidate All Register
RD_base	0x00C0-0x00C4	GICR_SYNCR	Redistributor Synchronize Register
SGI_base VLPI_base	0x0080-0x0070-0x0074	GICR_IGROUPR0 GICR_VPROPBASER	Interrupt Virtual Group Redistributor Properties Base Address Register 0
SGI_base VLPI_base	0x0100-0x0078-0x007C	GICR_ISENABLER0 GICR_VPENDBASER	Interrupt Virtual Set-Enable Redistributor LPI Pending Table Base Address Register 0
SGI_base	0x0180-0x0080	GICR_ICENABLER0 GICR_IGROUPR0	Interrupt Clear-Enable Group Register 0
SGI_base	0x0200-0x0100	GICR_ISPENDR0 GICR_ISENABLER0	Interrupt Set-Pending Set-Enable Register 0
SGI_base	0x0280-0x0180	GICR_ICPENDR0 GICR_ICENABLER0	Interrupt Clear-Pending Clear-Enable Register 0
SGI_base	0x0300-0x0200	GICR_ISACTIVER0 GICR_ISPENDR0	Interrupt Set-Active Set-Pending Register 0
SGI_base	0x0380-0x0280	GICR_ICACTIVER0 GICR_ICPENDR0	Interrupt Clear-Active Clear-Pending Register 0
SGI_base	0x0400 + 4n-0x0300	GICR_IPRIORITYR<n> GICR_ISACTIVER0	Interrupt Priority Set-Active Registers Register 0
SGI_base	0x0C00-0x0380	GICR_ICFGR0 GICR_ICACTIVER0	Interrupt Configuration Clear-Active Register 0
SGI_base	0x0C04-0x0400 + 4n	GICR_ICFGR1 GICR_IPRIORITYR<n>	Interrupt Configuration Priority Register 1 Registers
SGI_base	0x0D00-0x0C00	GICR_IGRPMODR0 GICR_ICFGR0	Interrupt Group Modifier Configuration Register 0
SGI_base	0x0E00-0x0C04	GICR_NSACR GICR_ICFGR1	Non-secure Interrupt Access Configuration Control Register Register 1
VLPI_base SGI_base	0x0070-0x0074-0x0D00	GICR_VPROPBASER GICR_IGRPMODR0	Virtual Interrupt Redistributor Group Properties Modifier Base Register Address Register 0
VLPI_base SGI_base	0x0078-0x007C-0x0E00	GICR_VPENDBASER GICR_NSACR	Virtual Non-secure Redistributor Access LPI Pending Table Base Address Control 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 + 4n	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 Virtual interfaceITS control block:

Offset	Name	Description
0x0000	GICH_HCR GITS_CTLR	HypervisorITS Control Register
0x0004	GICH_VTR GITS_IIDR	VirtualITS TypeIdentification Register
0x0008 0x0008-0x000C	GICH_VMCR GITS_TYPER	VirtualITS Machine ControlType Register
0x0010 0x0080-0x0084	GICH_MISR GITS_CBASER	MaintenanceITS InterruptCommand StatusQueue RegisterDescriptor
0x0020 0x0088-0x008C	GICH_EISR GITS_CWRITER	EndITS Interrupt StatusWrite Register
0x0030 0x0090-0x0094	GICH_ELRSR GITS_CREADR	EmptyITS List Register StatusRead Register
0x00F0 0x0100 + 4n 8n	GICH_APR<n> GITS_BASER<n>	ActiveITS PrioritiesTranslation RegistersTable-Descriptors
0x0100 + 4n	GICH_LR<n>	List Registers

In the PMUGICITS translation block:

Offset	Name	Description
0x000 + 8n 0x0040	PMEVCNTR<n>_EL0 GITS_TRANSLATER	PerformanceITS MonitorsTranslation Event Count RegistersRegister
0x0F8	PMCCNTR_EL0[31:0]	Performance Monitors Cycle Counter
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 + 4n	PMEVTYPER<n>_EL0	Performance Monitors Event Type Registers
0x47C	PMCCFILTR_EL0	Performance Monitors Cycle Counter Filter Register
0xC00	PMCNTENSET_EL0	Performance Monitors Count Enable Set register

Offset	Name	Description
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	PMOVSCLR_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
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 **TimerCTI** block:

Frame	Offset	Name	Description
CNTBaseN	0x000	CNTPCT[31:0] CTICONTROL	Counter-timerCTI PhysicalControl Countregister
CNTBaseN0x010	0x004	CNTPCT[63:32] CTHINTACK	Counter-timerCTI PhysicalOutput CountTrigger-Acknowledge register
CNTBaseN0x014	0x008	CNTVCT[31:0] CTIAPPSET	Counter-timerCTI VirtualApplication CountTrigger-Set register
CNTBaseN0x018	0x00C	CNTVCT[63:32] CTIAPPCLEAR	Counter-timerCTI VirtualApplication CountTrigger-Clear register
CNTBaseN0x01C	0x010	CNTRFQ CTIAPPPULSE	Counter-timerCTI FrequencyApplication Pulse register
CNTBaseN0x020+4n	0x014	CNTEL0ACR CTIHINEN<n>	Counter-timerCTI EL0Input AccessTrigger Controlto RegisterOutput-Channel Enable registers
CNTBaseN0x0A0+4n	0x018	CNTVOFF[31:0] CTIOUTEN<n>	Counter-timerCTI VirtualInput OffsetChannel-to-Output Trigger Enable registers
CNTBaseN0x130	0x01C	CNTVOFF[63:32] CTITRIGINSTATUS	Counter-timerCTI VirtualTrigger OffsetIn-Status register
CNTBaseN0x134	0x020	CNTP_CVAL[31:0] CTITRIGOUTSTATUS	Counter-timerCTI PhysicalTrigger TimerOut CompareValueStatus register
CNTBaseN0x138	0x024	CNTP_CVAL[63:32] CTICHINSTATUS	Counter-timerCTI PhysicalChannel TimerIn CompareValueStatus register

Frame	Offset	Name	Description
CNTBaseN0x13C	0x028	CNTP_TVAL CTICHOUTSTATUS	Counter-timerCTI PhysicalChannel TimerOut TimerValueStatus register
CNTBaseN0x140	0x02C	CNTP_CTL CTIGATE	Counter-timerCTI PhysicalChannel TimerGate ControlEnable register
CNTBaseN0x144	0x030	CNTV_CVAL[31:0] ASICCTL	Counter-timerCTI VirtualExternal TimerMultiplexer CompareValueControl register
CNTBaseN0xF00	0x034	CNTV_CVAL[63:32] CTHICTRL	Counter-timerCTI VirtualIntegration Timermode CompareValueControl register
CNTBaseN0xFA0	0x038	CNTV_TVAL CTICLAIMSET	Counter-timerCTI VirtualClaim TimerTag TimerValueSet register
CNTBaseN0xFA4	0x03C	CNTV_CTL CTICLAIMCLR	Counter-timerCTI VirtualClaim TimerTag ControlClear register
CNTBaseN0xFA8	0xFD0 + 4n	CounterID<n> CTIDEVAFF0	CounterCTI IDDevice registersAffinity-register 0
CNTCTLBase0xFAC	0x000	CNTFRQ CTIDEVAFF1	Counter-timerCTI FrequencyDevice Affinity-register 1
CNTCTLBase0xFB0	0x004	CNTNSAR CTILAR	Counter-timerCTI Non-secureLock Access Register
CNTCTLBase0xFB4	0x008	CNTTIDR CTILSR	Counter-timerCTI TimerLock IDStatus Register
CNTCTLBase0xFB8	0x040 + 4n	CNTACR<n> CTIAUTHSTATUS	Counter-timerCTI AccessAuthentication ControlStatus Registersregister
CNTCTLBase0xFBC	0x080 + 8n	CNTVOFF<n>[31:0] CTIDEVARCH	Counter-timerCTI VirtualDevice OffsetsArchitecture register
CNTCTLBase0xFC0	0x084 + 8n	CNTVOFF<n>[63:32] CTIDEVID2	Counter-timerCTI VirtualDevice OffsetsID-register 2
CNTCTLBase0xFC4	0xFD0 + 4n	CounterID<n> CTIDEVID1	CounterCTI Device-ID registersregister 1
CNTControlBase0xFC8	0x000	CNTCR CTIDEVID	CounterCTI ControlDevice RegisterID-register 0
CNTControlBase0xFCC	0x004	CNTSR CTIDEVTYPE	CounterCTI StatusDevice RegisterType register
CNTControlBase0xFD0	0x008	CNTCV[31:0] CTIPIDR4	CounterCTI CountPeripheral ValueIdentification registerRegister 4
CNTControlBase0xFE0	0x00C	CNTCV[63:32] CTIPIDR0	CounterCTI CountPeripheral ValueIdentification registerRegister 0
CNTControlBase0xFE4	0x020 + 4n	CNTFID<n> CTIPIDR1	CounterCTI FrequencyPeripheral IDs,Identification nRegister > 01
CNTControlBase0xFE8	0x020	CNTFID0 CTIPIDR2	CounterCTI FrequencyPeripheral IDIdentification-Register 2
CNTControlBase0xFEC	0xFD0 + 4n	CounterID<n> CTIPIDR3	CounterCTI IDPeripheral registersIdentification Register 3
CNTEL0BaseN0xFF0	0x000	CNTPCT[31:0] CTICIDR0	Counter-timerCTI PhysicalComponent CountIdentification Register 0
CNTEL0BaseN0xFF4	0x004	CNTPCT[63:32] CTICIDR1	Counter-timerCTI PhysicalComponent CountIdentification Register 1
CNTEL0BaseN0xFF8	0x008	CNTVCT[31:0] CTICIDR2	Counter-timerCTI VirtualComponent CountIdentification Register 2
CNTEL0BaseN0xFFC	0x00C	CNTVCT[63:32] CTICIDR3	Counter-timerCTI VirtualComponent CountIdentification Register 3
CNTEL0BaseN	0x010	CNTFRQ	Counter-timer Frequency
CNTEL0BaseN	0x020	CNTP_CVAL[31:0]	Counter-timer Physical Timer CompareValue
CNTEL0BaseN	0x024	CNTP_CVAL[63:32]	Counter-timer Physical Timer CompareValue
CNTEL0BaseN	0x028	CNTP_TVAL	Counter-timer Physical Timer TimerValue
CNTEL0BaseN	0x02C	CNTP_CTL	Counter-timer Physical Timer Control
CNTEL0BaseN	0x030	CNTV_CVAL[31:0]	Counter-timer Virtual Timer CompareValue
CNTEL0BaseN	0x034	CNTV_CVAL[63:32]	Counter-timer Virtual Timer CompareValue
CNTEL0BaseN	0x038	CNTV_TVAL	Counter-timer Virtual Timer TimerValue
CNTEL0BaseN	0x03C	CNTV_CTL	Counter-timer Virtual Timer Control

Frame	Offset	Name	Description
CNTELOBaseN	0xFD0 + 4n	CounterID<n>	Counter ID registers
CNTReadBase	0x000	CNTCV[31:0]	Counter Count Value register
CNTReadBase	0x004	CNTCV[63:32]	Counter Count Value register
CNTReadBase	0xFD0 + 4n	CounterID<n>	Counter ID registers

In the GIC Distributor block:

Offset	Name	Description
0x0000	GICD_CTLR	Distributor Control Register
0x0004	GICD_TYPER	Interrupt Controller Type Register
0x0008	GICD_HIDR	Distributor Implementer Identification Register
0x0010	GICD_STATUSR	Error Reporting Status Register
0x0040	GICD_SETSPI_NSR	Set Non-secure SPI Pending Register
0x0048	GICD_CLRSPI_NSR	Clear Non-secure SPI Pending Register
0x0050	GICD_SETSPI_SR	Set Secure SPI Pending Register
0x0058	GICD_CLRSPI_SR	Clear Secure SPI Pending Register
0x0080 + 4n	GICD_IGROUPR<n>	Interrupt Group Registers
0x0100 + 4n	GICD_ISENBALER<n>	Interrupt Set-Enable Registers
0x0180 + 4n	GICD_ICENABLER<n>	Interrupt Clear-Enable Registers
0x0200 + 4n	GICD_ISPENDR<n>	Interrupt Set-Pending Registers
0x0280 + 4n	GICD_ICPENDR<n>	Interrupt Clear-Pending Registers
0x0300 + 4n	GICD_ISACTIVER<n>	Interrupt Set-Active Registers
0x0380 + 4n	GICD_ICACTIVER<n>	Interrupt Clear-Active Registers
0x0400 + 4n	GICD_IPRIORITYR<n>	Interrupt Priority Registers
0x0800 + 4n	GICD_ITARGETSR<n>	Interrupt Processor Targets Registers
0x0C00 + 4n	GICD_ICFGR<n>	Interrupt Configuration Registers
0x0D00 + 4n	GICD_IGRPMODR<n>	Interrupt Group Modifier Registers
0x0E00 + 4n	GICD_NSACR<n>	Non-secure Access Control Registers
0x0F00	GICD_SGIR	Software Generated Interrupt Register
0x0F10 + 4n	GICD_CPENDSGIR<n>	SGI Clear-Pending Registers
0x0F20 + 4n	GICD_SPENDSGIR<n>	SGI Set-Pending Registers
0x6000 + 8n	GICD_IROUTER<n>	Interrupt Routing Registers

28/0907/2017 0816:4140

Copyright Â© 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTFID<n>, Counter Frequency IDs, n > 0

The CNTFID<n> characteristics are:

Purpose

Indicates alternative system counter update frequencies.
This register is part of the Generic Timer registers functional group.

Usage constraints

This register is accessible as follows:

Default
RO or RW

In a system that supports Secure and Non-secure memory maps the CNTControlBase frame, that includes these registers, is implemented only in the Secure memory map.

Configuration

The power domain of CNTFID<n> is IMPLEMENTATION DEFINED.

If this register is implemented as an RW register, on a reset of the reset domain in which it is implemented, RW fields in this register reset to UNKNOWN values. The register is not affected by a reset of any other reset domain. For more information see 'Power and reset domains for the system level implementation of the Generic Timer' in Chapter I1 of the ARMv8 ARM.

The possible frequencies for the system counter are stored in the Frequency modes table as 32-bit words starting with the base frequency, [CNTFID0](#), see 'The Frequency modes table' in Chapter I1 of the ARMv8 ARM.

The number of CNTFID<n> registers is IMPLEMENTATION DEFINED, and the only required CNTFID<n> register is [CNTFID0](#).

The final entry in the Frequency modes table must be followed by a 32-bit word of zero value, to mark the end of the table.

Typically, the Frequency modes table will be in read-only memory. However, a system implementation might use read/write memory for the table, and initialize the table entries as part of its start-up sequence.

If the Frequency modes table is in read/write memory, ARM strongly recommends that the table is not updated once the system is running.

Attributes

CNTFID<n> is a 32-bit register.

Field descriptions

The CNTFID<n> bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
																Frequency															

Frequency, bits [31:0]

A system counter update frequency, in Hz. Must be an exact divisor of the base frequency. ARM strongly recommends that all frequency values in the Frequency modes table are integer power-of-two divisors of the base frequency.

When the system timer is operating at a lower frequency than the base frequency, the increment applied at each counter update is given by:

$\text{increment} = (\text{base frequency}) / (\text{selected frequency})$

Accessing the CNTFID<n>

CNTFID<n> can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTControlBase	$0 \times 020 + 4n$

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

<u>SysReg_v83A_xml-00bet4</u> <u>(old)</u>	htmldiff from-	<u>(new)</u> <u>SysReg_v83A_xml-00bet5</u>
---	----------------	---

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTP_CVAL, Counter-timer Physical Timer CompareValue

The CNTP_CVAL characteristics are:

Purpose

Holds the 64-bit compare value for the EL1 physical timer.

This register is part of the Generic Timer registers functional group.

Usage constraints

This register is accessible as follows:

Default
RW

CNTP_CVAL can be implemented in any implemented CNTBaseN frame, and in the corresponding CNTEL0BaseN frame.

'CNTCTLBase status and control fields for the CNTBaseN and CNTEL0BaseN frames' in Chapter I1 of the ARMv8 ARM 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.

For an implemented CNTBaseN frame:

- CNTP_CVAL is accessible in that frame if the value of [CNTACR<n>.RWPT](#) is 1.
- Otherwise, the CNTP_CVAL address in that frame is RAZ/WI.

For an implemented CNTEL0BaseN frame:

- CNTP_CVAL is accessible in that frame if both:
 - CNTP_CVAL is accessible in the corresponding CNTBaseN frame:
 - The value of [CNTEL0ACR.EL0PTEN](#) is 1.
- Otherwise, the CNTP_CVAL address in that frame is RAZ/WI.

If the implementation supports 64-bit atomic accesses, then the CNTP_CVAL register must be accessible as an atomic 64-bit value.

Configuration

The power domain of CNTP_CVAL is IMPLEMENTATION DEFINED.

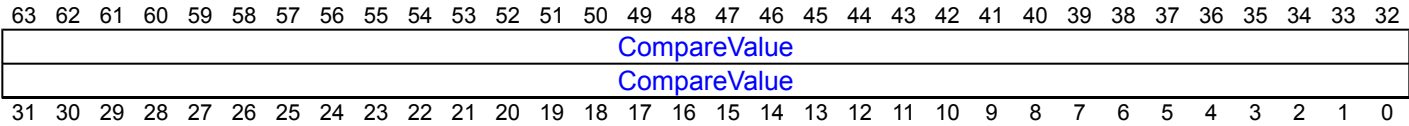
On a reset of the reset domain in which an RW instance of this register is implemented, RW fields in the register reset to UNKNOWN values. The register is not affected by a reset of any other reset domain. For more information see 'Power and reset domains for the system level implementation of the Generic Timer' in Chapter I1 of the ARMv8 ARM.

Attributes

CNTP_CVAL is a 64-bit register.

Field descriptions

The CNTP_CVAL bit assignments are:



CompareValue, bits [63:0]

Holds the EL1 physical timer CompareValue.

When [CNTP_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:

- [CNTP_CTL](#).ISTATUS is set to 1.
- An interrupt is generated if [CNTP_CTL](#).IMASK is 0.

When [CNTP_CTL](#).ENABLE is 0, the timer condition is not met, but [CNTPCT](#) continues to count.

Accessing the CNTP_CVAL

CNTP_CVAL[31:0] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x020
Timer	CNTELOBaseN	0x020

CNTP_CVAL[63:32] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x024
Timer	CNTELOBaseN	0x024

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg_v83A_xml-00bet4 (old)	htmldiff from-	(new)
	SysReg_v83A_xml-00bet4	SysReg_v83A_xml-00bet5

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTP_TVAL, Counter-timer Physical Timer TimerValue

The CNTP_TVAL characteristics are:

Purpose

Holds the timer value for the EL1 physical timer.

This register is part of the Generic Timer registers functional group.

Usage constraints

This register is accessible as follows:

Default
RW

CNTP_TVAL can be implemented in any implemented CNTBaseN frame, and in the corresponding CNTEL0BaseN frame.

'CNTCTLBase status and control fields for the CNTBaseN and CNTEL0BaseN frames' in Chapter I1 of the ARMv8 ARM 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.

For an implemented CNTBaseN frame:

- CNTP_TVAL is accessible in that frame if the value of [CNTACR<n>.RWPT](#) is 1.
- Otherwise, the CNTP_TVAL address in that frame is RAZ/WI.

For an implemented CNTEL0BaseN frame:

- CNTP_TVAL is accessible in that frame if both:
 - CNTP_TVAL is accessible in the corresponding CNTBaseN frame:
 - The value of [CNTEL0ACR.EL0PTEN](#) is 1.
- Otherwise, the CNTP_TVAL address in that frame is RAZ/WI.

Configuration

The power domain of CNTP_TVAL is IMPLEMENTATION DEFINED.

On a reset of the reset domain in which an RW instance of this register is implemented, RW fields in the register reset to UNKNOWN values. The register is not affected by a reset of any other reset domain. For more information see 'Power and reset domains for the system level implementation of the Generic Timer' in Chapter I1 of the ARMv8 ARM.

Attributes

CNTP_TVAL is a 32-bit register.

Field descriptions

The CNTP_TVAL bit assignments are:

31	30	29	28	27	26	25	24	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 (CompareValue - [CNTPCT](#)).

On a write of this register, CompareValue 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](#) - CompareValue) 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.

Accessing the CNTP_TVAL

CNTP_TVAL can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x028
Timer	CNTEL0BaseN	0x028

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTV_CVAL, Counter-timer Virtual Timer CompareValue

The CNTV_CVAL characteristics are:

Purpose

Holds the 64-bit compare value for the virtual timer.

This register is part of the Generic Timer registers functional group.

Usage constraints

This register is accessible as follows:

Default
RW

CNTV_CVAL can be implemented in any implemented CNTBaseN frame that has virtual timer capability, and in the corresponding CNTEL0BaseN frame.

'CNTCTLBase status and control fields for the CNTBaseN and CNTEL0BaseN frames' in Chapter I1 of the ARMv8 ARM 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.

For an implemented CNTBaseN frame that has virtual timer capability:

- CNTV_CVAL is accessible in that frame if the value of [CNTACR<n>.RWVT](#) is 1.
- Otherwise, the CNTV_CVAL address in that frame is RAZ/WI.

For an implemented CNTEL0BaseN frame:

- CNTV_CVAL is accessible in that frame if both:
 - CNTV_CVAL is accessible in the corresponding CNTBaseN frame:
 - The value of [CNTEL0ACR.EL0VTEN](#) is 1.
- Otherwise, the CNTV_CVAL address in that frame is RAZ/WI.

If the implementation supports 64-bit atomic accesses, then the CNTV_CVAL register must be accessible as an atomic 64-bit value.

Configuration

The power domain of CNTV_CVAL is IMPLEMENTATION DEFINED.

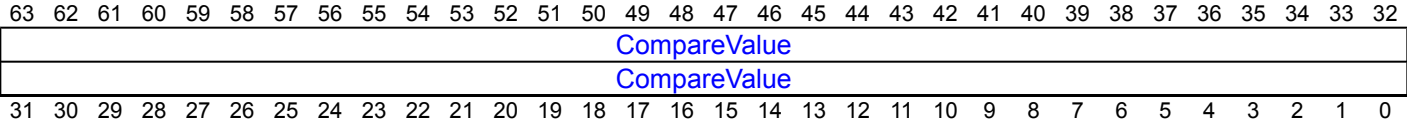
On a reset of the reset domain in which an RW instance of this register is implemented, RW fields in the register reset to UNKNOWN values. The register is not affected by a reset of any other reset domain. For more information see 'Power and reset domains for the system level implementation of the Generic Timer' in Chapter I1 of the ARMv8 ARM.

Attributes

CNTV_CVAL is a 64-bit register.

Field descriptions

The CNTV_CVAL bit assignments are:



CompareValue, bits [63:0]

Holds the virtual timer CompareValue.

When CNTV_CTL.ENABLE is 1, the timer condition is met when (CNTVCT - 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:

- CNTV_CTL.ISTATUS is set to 1.
- An interrupt is generated if CNTV_CTL.IMASK is 0.

When CNTV_CTL.ENABLE is 0, the timer condition is not met, but CNTVCT continues to count.

Accessing the CNTV_CVAL

CNTV_CVAL[31:0] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x030
Timer	CNTELOBaseN	0x030

CNTV_CVAL[63:32] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x034
Timer	CNTELOBaseN	0x034

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

<u>SysReg_v83A_xml-00bet4</u> <u>(old)</u>	htmldiff from-	<u>(new)</u>
	SysReg_v83A_xml-00bet4	<u>SysReg_v83A_xml-00bet5</u>

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

CNTV_TVAL, Counter-timer Virtual Timer TimerValue

The CNTV_TVAL characteristics are:

Purpose

Holds the timer value for the virtual timer.

This register is part of the Generic Timer registers functional group.

Usage constraints

This register is accessible as follows:

Default
RW

CNTV_TVAL can be implemented in any implemented CNTBaseN frame that has virtual timer capability, and in the corresponding CNTEL0BaseN frame.

'CNTCTLBase status and control fields for the CNTBaseN and CNTEL0BaseN frames' in Chapter I1 of the ARMv8 ARM 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.

For an implemented CNTBaseN frame that has virtual timer capability:

- CNTV_TVAL is accessible in that frame if the value of [CNTACR<n>.RWVT](#) is 1.
- Otherwise, the CNTV_TVAL address in that frame is RAZ/WI.

For an implemented CNTEL0BaseN frame:

- CNTV_TVAL is accessible in that frame if both:
 - CNTV_TVAL is accessible in the corresponding CNTBaseN frame:
 - The value of [CNTEL0ACR.EL0VTEN](#) is 1.
- Otherwise, the CNTV_TVAL address in that frame is RAZ/WI.

Configuration

The power domain of CNTV_TVAL is IMPLEMENTATION DEFINED.

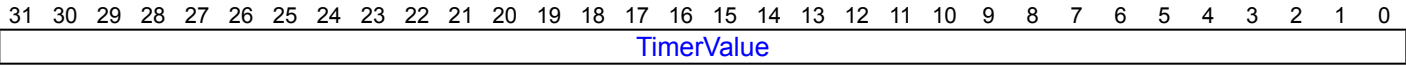
On a reset of the reset domain in which an RW instance of this register is implemented, RW fields in the register reset to UNKNOWN values. The register is not affected by a reset of any other reset domain. For more information see 'Power and reset domains for the system level implementation of the Generic Timer' in Chapter I1 of the ARMv8 ARM.

Attributes

CNTV_TVAL is a 32-bit register.

Field descriptions

The CNTV_TVAL bit assignments are:



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 (CompareValue - CNTVCT0) - CNTVCT).

On a write of this register, CompareValue is set to (CNTVCT0 + TimerValue), where TimerValue is treated as a signed 32-bit integer. CNTVCT + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When CNTV_CTL.ENABLE is 1, the timer condition is met when (CNTVCT - CompareValue) 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.

Accessing the CNTV_TVAL

CNTV_TVAL can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x038
Timer	CNTELOBaseN	0x038

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg_v83A_xml-00bet4
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
SysReg_v83A_xml-00bet5

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

EDPRSR, External Debug Processor Status Register

The EDPRSR characteristics are:

Purpose

Holds information about the reset and powerdown state of the PE.

This register is part of the Debug registers functional group.

Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

On permitted accesses to the register, other access controls affect the behavior of some fields. See the field descriptions for more information.

If the Core power domain is powered up (EDPRSR.PU == 1), then following a read of EDPRSR:

- If DoubleLockStatus() == FALSE, then:
 - EDPRSR.{SDR, SPMAD, SDAD, SPD} are cleared to 0.
 - EDPRSR.SR is cleared to 0 if the non-debug logic of the PE is not in reset state (EDPRSR.R == 0).
- Otherwise it is CONSTRAINED UNPREDICTABLE whether or not this clearing occurs.

If the Core power domain is powered down (EDPRSR.PU == 0), then:

- EDPRSR.{SDR, SPMAD, SDAD, SR} are all UNKNOWN, and are either reset or restored on being powered up.
- EDPRSR.SPD is not cleared following a read of EDPRSR. See the SPD bit description for more information.

The clearing of bits is an indirect write to EDPRSR.

Configuration

EDPRSR contains fields that are in the Core power domain and fields that are in the Debug power domain.

Some of the fields in the Core power domain are in the Cold reset domain and others are in the Warm reset domain. See the field descriptions for more information. However:

- Fields that are in the Cold reset domain are not affected by a warm reset and are not affected by an External debug reset.
- Fields in the Warm reset domain are also reset by a Cold reset but are not affected by an External debug reset.
- Fields in the Debug power domain are not affected by a Warm reset and are not affected by a Cold reset.

Attributes

EDPRSR is a 32-bit register.

Field descriptions

The EDPRSR bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SDR	SPMAD	SPMAD	SDAD	EDAD	DLK	OSLK	HALTED	SR	R	SPD	PU

Bits [31:12]

Reserved, RES0.

SDR, bit [11]

Sticky debug restart. Set to 1 when the PE exits Debug state.

This bit is UNKNOWN on reads if any of the following are true:

- DoubleLockStatus() == TRUE. The OS double-lock is locked.
- EDPRSR.R == 1. The PE is in Reset state.
- EDPRSR.PU == 0. The Core power domain is powered down.

Otherwise permitted values are:

SDR	Meaning
0	The PE has not restarted since EDPRSR was last read.
1	The PE has restarted since EDPRSR was last read.

Note

If a reset occurs when the PE is in Debug state, the PE exits Debug state. SDR is UNKNOWN on Warm reset, meaning a debugger must also use the SR bit to determine whether the PE has left Debug state.

If EDPRSR.PU reads as 1, which means that the Core power domain is in a powerup state, then following a read of EDPRSR:

- If DoubleLockStatus() == FALSE this bit clears to 0.
- If DoubleLockStatus() == TRUE, it is CONSTRAINED UNPREDICTABLE whether this bit clears to 0 or is unchanged.

This field is in the Core power domain and the Warm reset domain. On a Warm or Cold reset it resets to an UNKNOWN value.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

This field resets to its defined reset value on Warm reset.

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	SLK	Default
UNK	UNK	RO	RC

'Access permissions for the External debug interface registers' in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*, section H8.6.1 describes the conditions shown in this table. These conditions are prioritized, with the leftmost condition having the highest priority and priority decreasing from left to right.

SPMAD, bit [10]

Sticky EPMAD error. Set to 1 if an external debug interface access to a Performance Monitors register returns an error because AllowExternalPMUAccess() == FALSE.

This bit is UNKNOWN on reads if any of the following are true:

- DoubleLockStatus() == TRUE
- Either of EDPRSR.{OSLK, R} is set to 1.
- EDPRSR.PU is 0.

Otherwise permitted values are:

SPMAD	Meaning
0	No accesses to the external Performance Monitors registers have failed since EDPRSR was last read.
1	At least one access to the external Performance Monitors registers has failed since EDPRSR was last read.

If the Core power domain is powered up, then, following a read of EDPRSR:

- If DoubleLockStatus() == FALSE this bit clears to 0.
- If DoubleLockStatus() == TRUE, it is CONSTRAINED UNPREDICTABLE whether this bit clears to 0 or is unchanged.

The write to SPMAD is an indirect write to EDPRSR that is a side effect of the access. The indirect write might not occur for a memory-mapped access to the external debug interface.

This field is in the Core power domain and the Cold reset domain. On a Cold reset it resets to 0.

When this register has an architecturally-defined reset value, this field resets to 0.

This field resets to its defined reset value on Cold reset.

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	OSLK	SLK	Default
UNK	UNK	UNK	RO	RC

'Access permissions for the External debug interface registers' in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*, section H8.6.1 describes the conditions shown in this table. These conditions are prioritized, with the leftmost condition having the highest priority and priority decreasing from left to right.

EPMAD, bit [9]

External Performance Monitors access disable status.

This bit is UNKNOWN on reads if any of the following is true:

- DoubleLockStatus() == TRUE
- Either of EDPRSR.{OSLK, R} is set to 1.
- EDPRSR.PU is 0.

Otherwise permitted values are:

EPMAD	Meaning
0	External Performance Monitors access enabled. AllowExternalPMUAccess() == TRUE.
1	External Performance Monitors access disabled. AllowExternalPMUAccess() == FALSE.

If external performance monitors access is not implemented, EPMAD is RAO.

This field is in the Core power domain.

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	OSLK	EPMAD	Default
UNK	UNK	UNK	RAO	RO

'Access permissions for the External debug interface registers' in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*, section H8.6.1 describes the conditions shown in this table. These conditions are prioritized, with the leftmost condition having the highest priority and priority decreasing from left to right.

SDAD, bit [8]

Sticky EDAD error. Set to 1 if an external debug interface access to a debug register returns an error because AllowExternalDebugAccess() == FALSE.

This bit is UNKNOWN on reads if any of the following are true:

- DoubleLockStatus() == TRUE
- EDPRSR.R is 1.
- EDPRSR.PU is 0.
- EDPRSR.OSLK is 1 and external debug writes to [OSLAR_EL1](#) do not return an error when AllowExternalDebugAccess() == FALSE.

Otherwise permitted values are:

SDAD	Meaning
0	No accesses to the external debug registers have failed since EDPRSR was last read.
1	At least one access to the external debug registers has failed since EDPRSR was last read.

If the Core power domain is powered up, then, following a read of EDPRSR:

- If DoubleLockStatus() == FALSE this bit clears to 0.
- If DoubleLockStatus() == TRUE, it is CONSTRAINED UNPREDICTABLE whether this bit clears to 0 or is unchanged.

The write to SDAD is an indirect write to EDPRSR that is a side effect of the access. The indirect write might not occur for a memory-mapped access to the external debug interface.

This field is in the Core power domain and the Cold reset domain. On a Cold reset it resets to 0.

When this register has an architecturally-defined reset value, this field resets to 0.

This field resets to its defined reset value on Cold reset.

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	OSLK	SLK	Default
UNK	UNK	See text	RO	RC

'Access permissions for the External debug interface registers' in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*, section H8.6.1 describes the conditions shown in this table. These conditions are prioritized, with the leftmost condition having the highest priority and priority decreasing from left to right.

EDAD, bit [7]

External debug access disable status.

This bit is UNKNOWN on reads if any of the following are true:

- DoubleLockStatus() == TRUE
- EDPRSR.R is 1.
- EDPRSR.PU is 0.
- EDPRSR.OSLK is 1 and external debug writes to [OSLAR_EL1](#) do not return an error when AllowExternalDebugAccess() == FALSE.

Otherwise permitted values are:

EDAD	Meaning
0	External debug access enabled. AllowExternalDebugAccess() == TRUE.
1	External debug access disabled. AllowExternalDebugAccess() == FALSE.

This field is in the Core power domain.

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	OSLK	EDAD	Default
UNK	UNK	See text	RAO	RAZ

'Access permissions for the External debug interface registers' in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*, section H8.6.1 describes the conditions shown in this table. These conditions are prioritized, with the leftmost condition having the highest priority and priority decreasing from left to right.

DLK, bit [6]

OS Double Lock status bit. Returns the result of the pseudocode function DoubleLockStatus().

This bit is UNKNOWN on reads if EDPRSR.PU is 0.

Otherwise reads as zero if any of the following are true, that is when DoubleLockStatus() == FALSE:

- [OSDLR_EL1](#).DLK == 0.
- [DBGPRCR_EL1](#).CORENPDRQ == 1.

- The PE is in Debug state.

In ARMv8.0 and ARMv8.1, if the Core power domain is powered up and DoubleLockStatus() == TRUE, it is IMPLEMENTATION DEFINED whether:

- EDPRSR.PU reads as 1, EDPRSR.DLK reads as 1, and EDPRSR.SPD is UNKNOWN.
- EDPRSR.PU reads as 0, EDPRSR.DLK is UNKNOWN, and EDPRSR.SPD reads as 0.

From ARMv8.2, if the Core power domain is powered up and DoubleLockStatus() == TRUE, then EDPRSR.PU reads as 0, EDPRSR.DLK is UNKNOWN, and EDPRSR.SPD reads as 0.

If the Core power domain is powered up and entered reset state with the OS double-lock locked this bit has a CONSTRAINED UNPREDICTABLE value, for more information see 'EDPRSR.{DLK, R} and reset state' in the ARMv8 ARM, section H6 (Debug Reset and Powerdown Support)

EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see 'EDPRSR.{DLK, SPD, PU} bits record accessibility and lost of state in Core power domain' in the ARMv8 ARM, section H6 (Debug Reset and Powerdown Support)

Note

Use of this bit by debuggers is deprecated from ARMv8.2.

This field is in the Core power domain.

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	Default
UNK	See text	RAZ

'Access permissions for the External debug interface registers' in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*, section H8.6.1 describes the conditions shown in this table. These conditions are prioritized, with the leftmost condition having the highest priority and priority decreasing from left to right.

OSLK, bit [5]

OS lock status bit.

This bit is UNKNOWN on reads if either:

- DoubleLockStatus() == TRUE
- EDPRSR.R is 1.
- EDPRSR.PU is 0.

A read of this bit returns the value of [OSLSR_EL1.OSLK](#).

This field is in the Core power domain.

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	OSLK	Default
UNK	UNK	RAO	RAZ

'Access permissions for the External debug interface registers' in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*, section H8.6.1 describes the conditions shown in this table. These conditions are prioritized, with the leftmost condition having the highest priority and priority decreasing from left to right.

HALTED, bit [4]

Halted status bit.

This bit is UNKNOWN on reads if EDPRSR.PU is 0.

Otherwise permitted values are:

HALTED	Meaning
0	PE is in Non-debug state.
1	PE is in Debug state.

Because the OS Double Lock is never set when the PE is in Debug state, this bit is always RAZ when DoubleLockStatus() == TRUE.

This field is in the Core power domain.

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	Default
UNK	See text	RO

'Access permissions for the External debug interface registers' in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*, section H8.6.1 describes the conditions shown in this table. These conditions are prioritized, with the leftmost condition having the highest priority and priority decreasing from left to right.

SR, bit [3]

Sticky core reset status bit.

This bit is UNKNOWN on reads if either:

- DoubleLockStatus() == TRUE
- EDPRSR.PU is 0.

Otherwise permitted values are:

SR	Meaning
0	The non-debug logic of the PE is not in reset state and has not been reset since the last time EDPRSR was read.
1	The non-debug logic of the PE is in reset state or has been reset since the last time EDPRSR was read.

If EDPRSR.PU reads as 1 and EDPRSR.R reads as 0, which means that the Core power domain is in a powerup state and that the non-debug logic of the PE is not in reset state, then following a read of EDPRSR:

- If DoubleLockStatus() == FALSE this bit clears to 0.
- If DoubleLockStatus() == TRUE, it is UNPREDICTABLE whether this bit clears to 0 or is unchanged.

This field is in the Core power domain and the Warm reset domain. On a Warm or Cold reset it resets to 1.

When this register has an architecturally-defined reset value, this field resets to 1.

This field resets to its defined reset value on Warm reset.

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	SLK	Default
UNK	UNK	RO	RC

'Access permissions for the External debug interface registers' in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*, section H8.6.1 describes the conditions shown in this table. These conditions are prioritized, with the leftmost condition having the highest priority and priority decreasing from left to right.

R, bit [2]

PE reset status bit.

This bit is UNKNOWN on reads if either:

- DoubleLockStatus() == TRUE
- EDPRSR.PU is 0.

Otherwise permitted values are:

R	Meaning
0	The non-debug logic of the PE is not in reset state.
1	The non-debug logic of the PE is in reset state.

If the Core power domain is powered up and entered reset state with the OS double-lock locked this bit has a CONSTRAINED UNPREDICTABLE value, for more information see 'EDPRSR.{DLK, R} and reset state' in the ARMv8 ARM, section H6 (Debug Reset and Powerdown Support)

This field is in the Core power domain.

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	Default
UNK	UNK	RO

'Access permissions for the External debug interface registers' in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*, section H8.6.1 describes the conditions shown in this table. These conditions are prioritized, with the leftmost condition having the highest priority and priority decreasing from left to right.

SPD, bit [1]

Sticky core powerdown status bit.

This bit is UNKNOWN on reads if EDPRSR.PU is 1 and DoubleLockStatus() == TRUE .

Otherwise, permitted values are:

SPD	Meaning
0	If EDPRSR.PU is 0, it is not known whether the state of the debug registers in the Core power domain is lost. If EDPRSR.PU is 1, the state of the debug registers in the Core power domain has not been lost.
1	The state of the debug registers in the Core power domain has been lost.

If the Core power domain is powered up, then, following a read of EDPRSR:

- If DoubleLockStatus() == FALSE this bit clears to 0.
- If DoubleLockStatus() == TRUE, it is CONSTRAINED UNPREDICTABLE whether this bit clears to 0 or is unchanged.

When the value of EDPRSR.PU is 0 indicating that the Core power domain is in either retention or powerdown state, EDPRSR.SPD reads as 0. For more information, see 'EDPRSR.SPD when the Core domain is in either retention or powerdown state' in the ARMv8 ARM, section H6 (Debug Reset and Powerdown Support).

EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see 'EDPRSR.{DLK, SPD, PU} bits record accessibility and lost of state in Core power domain' in the ARMv8 ARM, section H6 (Debug Reset and Powerdown Support).

This field is in the Core power domain and the Cold reset domain. On a Cold reset it resets to 1.

When this register has an architecturally-defined reset value, this field resets to 1.

This field resets to its defined reset value on Cold reset.

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	SLK	Default
RO	UNK	RO	RC

'Access permissions for the External debug interface registers' in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*, section H8.6.1 describes the conditions shown in this table. These conditions are prioritized, with the leftmost condition having the highest priority and priority decreasing from left to right.

PU, bit [0]

Core powerup status bit. Indicates whether the Core power domain debug registers can be accessed.

When the Core power domain is powered-up and OS double-lock is locked, then:

- When ARMv8.2-Debug is implemented, the value of EDPRSR.PU reads as 0.
- When ARMv8.2-Debug is not implemented, the value of EDPRSR.PU is IMPLEMENTATION DEFINED.

See the description of DLK for more information.

Otherwise, permitted values are:

PU	Meaning
0	Core is in a low-power or powerdown state where the debug registers cannot be accessed.
1	Core is in a powerup state where the debug registers can be accessed.

If the Core power domain is powered up and entered reset state with the OS double-lock locked this bit has a CONSTRAINED UNPREDICTABLE value, for more information see 'EDPRSR.{DLK, R} and reset state' in the ARMv8 ARM, section H6 (Debug Reset and Powerdown Support)

EDPRSR.{DLK, SPD, PU} describe whether registers in the Core power domain can be accessed, and whether their state has been lost since the last time the register was read. For more information, see 'EDPRSR.{DLK, SPD, PU} bits record accessibility and lost of state in Core power domain' in the ARMv8 ARM, section H6 (Debug Reset and Powerdown Support)

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	Default
RAZ	See text	RAO

'Access permissions for the External debug interface registers' in the *ARM[®] Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*, section H8.6.1 describes the conditions shown in this table. These conditions are prioritized, with the leftmost condition having the highest priority and priority decreasing from left to right.

Accessing the EDPRSR

EDPRSR can be accessed through the external debug interface:

Component	Offset
Debug	0x314

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

MIDR_EL1, Main ID Register

The MIDR_EL1 characteristics are:

Purpose

Provides identification information for the PE, including an implementer code for the device and a device ID number.

This register is part of the Identification registers functional group.

Usage constraints

This register is accessible as follows:

Off	DLK	Default
IMP DEF	IMP DEF	RO

Configuration

External register MIDR_EL1 is architecturally mapped to AArch64 System register [MIDR_EL1](#).

External register MIDR_EL1 is architecturally mapped to AArch32 System register [MIDR](#).

It is IMPLEMENTATION DEFINED whether MIDR_EL1 is implemented in the Core power domain or in the Debug power domain.

Attributes

MIDR_EL1 is a 32-bit register.

Field descriptions

The MIDR_EL1 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Implementer								Variant				Architecture				PartNum								Revision							

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:

Hex representation	ASCII representation	Implementer
0x41	A	ARM Limited
0x42	B	Broadcom Corporation
0x43	C	Cavium Inc.
0x44	D	Digital Equipment Corporation
0x49	I	Infineon Technologies AG
0x4D	M	Motorola or Freescale Semiconductor Inc.
0x4E	N	NVIDIA Corporation
0x50	P	Applied Micro Circuits Corporation
0x51	Q	Qualcomm Inc.
0x56	V	Marvell International Ltd.
0x69	i	Intel Corporation

ARM can assign codes that are not published in this manual. All values not assigned by ARM are reserved and must not be used.

Variant, bits [23:20]

An IMPLEMENTATION DEFINED variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product.

Architecture, bits [19:16]

The permitted values of this field are:

Architecture	Meaning
0001	ARMv4
0010	ARMv4T
0011	ARMv5 (obsolete)
0100	ARMv5T
0101	ARMv5TE
0110	ARMv5TEJ
0111	ARMv6
1111	Architectural features are individually identified in the ID_* registers, see 'ID Identification registers, functional group' in the ARMv8 ARM, section K12.7.2G4.18.1.

All other values are reserved.

PartNum, bits [15:4]

An IMPLEMENTATION DEFINED primary part number for the device.

On processors implemented by ARM, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently.

Revision, bits [3:0]

An IMPLEMENTATION DEFINED revision number for the device.

Accessing the MIDR_EL1

MIDR_EL1 can be accessed through the external debug interface:

Component	Offset
Debug	0xD00

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMCEID0, Performance Monitors Common Event Identification register 0

The PMCEID0 characteristics are:

Purpose

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x000 to 0x01F.

When the value of a bit in the register is 1 the corresponding common event is implemented or counted.

Note

ARM recommends that, if a **common** event is never counted, the value of the corresponding register bit is 0.

For more information about the common events and the use of the PMCEIDn registers see The section describing 'Event numbers and common events' in chapter D5 'The Performance Monitors Extension' of the ARM Architecture Reference Manual, for ARMv8-A architecture profile.

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x000 to 0x01F.

Note

- ARM recommends that, if a **common** event is never counted, the value of the corresponding register bit is 0.
- This view of the register was previously called PMCEID0_EL0.

This register is part of the Performance Monitors registers functional group.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EPMAD	SLK	Default
Error	Error	Error	Error	RO	RO

Configuration

External register PMCEID0 is architecturally mapped to AArch64 System register [PMCEID0_EL0\[31:0\]](#).

External register PMCEID0 bits [31:0] are architecturally mapped to AArch32 System register [PMCEID0](#).

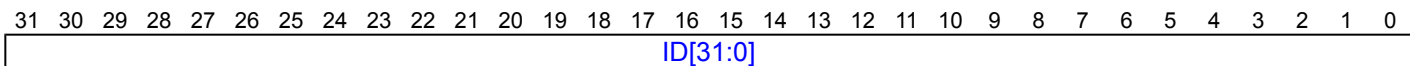
PMCEID0 is in the Core power domain.

Attributes

PMCEID0 is a 32-bit register.

Field descriptions

The PMCEID0 bit assignments are:

**ID[31:0], bits [31:0]**

ID[n] corresponds to common event n.

For each bit:

ID[31:0]	Meaning
0	The common event is not implemented, or not counted.
1	The common event is implemented.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing the PMCEID0

PMCEID0 can be accessed through the external debug interface:

Component	Offset
PMU	0xE20

28/09/2017 08:16:24

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

<u>SysReg_v83A_xml-00bet4</u> (old)	htmldiff from-	(new)
	SysReg_v83A_xml-00bet4	<u>SysReg_v83A_xml-00bet5</u>

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMCEID1, Performance Monitors Common Event Identification register 1

The PMCEID1 characteristics are:

Purpose

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x020 to 0x03F.

When the value of a bit in the register is 1 the corresponding common event is implemented or counted.

Note

ARM recommends that, if a **common** event is never counted, the value of the corresponding register bit is 0.

For more information about the common events and the use of the PMCEIDn registers see The section describing 'Event numbers and common events' in chapter D5 'The Performance Monitors Extension' of the ARM Architecture Reference Manual, for ARMv8-A architecture profile.

Note

- ARM recommends that, if a **common** event is never counted, the value of the corresponding register bit is 0.
- This view of the register was previously called PMCEID1_EL0.

This register is part of the Performance Monitors registers functional group.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EPMAD	SLK	Default
Error	Error	Error	Error	RO	RO

Configuration

External register PMCEID1 is architecturally mapped to AArch64 System register [PMCEID1_EL0\[31:0\]](#).

External register PMCEID1 bits [31:0] are architecturally mapped to AArch32 System register [PMCEID1](#).

PMCEID1 is in the Core power domain.

Attributes

PMCEID1 is a 32-bit register.

Field descriptions

The PMCEID1 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[63:32]																															

ID[63:32], bits [31:0]

ID[n] corresponds to common event n.

For each bit:

ID[63:32]	Meaning
0	The common event is not implemented, or not counted.
1	The common event is implemented.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing the PMCEID1

PMCEID1 can be accessed through the external debug interface:

Component	Offset
PMU	0xE24

28/09/2017 08:16:24

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
[\(old\)](#)

htmldiff from-
SysReg_v83A_xml-00bet4

[\(new\)](#)
[SysReg_v83A_xml-00bet5](#)

PMCEID2, Performance Monitors Common Event Identification register 2

The PMCEID2 characteristics are:

Purpose

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x4000 to 0x401F.

When the value of a bit in the register is 1 the corresponding common event is implemented or counted.

Note

ARM recommends that, if a **common** event is never counted, the value of the corresponding register bit is 0.

For more information about the common events and the use of the PMCEIDn registers see The section describing 'Event numbers and common events' in chapter D5 'The Performance Monitors Extension' of the ARM Architecture Reference Manual, for ARMv8-A architecture profile.

This register is part of the Performance Monitors registers functional group.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EPMAD	SLK	Default
Error	Error	Error	Error	RO	RO

Configuration

External register PMCEID2 is architecturally mapped to AArch64 System register [PMCEID0_EL0\[63:32\]](#).

External register PMCEID2 bits [63:32] are architecturally mapped to AArch32 System register [PMCEID2](#).

PMCEID2 is in the Core power domain.

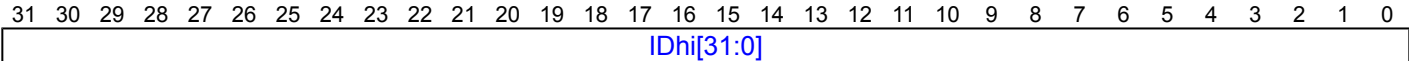
This register is introduced in ARMv8.1.

Attributes

PMCEID2 is a 32-bit register.

Field descriptions

The PMCEID2 bit assignments are:



IDhi[31:0], bits [31:0]

IDhi[n] corresponds to common event (0x4000 + n).

For each bit:

IDhi[31:0]	Meaning
0	The common event is not implemented, or not counted.
1	The common event is implemented.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing the PMCEID2

PMCEID2 can be accessed through the external debug interface:

Component	Offset
PMU	0xE28

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMCEID3, Performance Monitors Common Event Identification register 3

The PMCEID3 characteristics are:

Purpose

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x4020 to 0x403F.

When the value of a bit in the register is 1 the corresponding common event is implemented or counted.

Note

ARM recommends that, if a **common** event is never counted, the value of the corresponding register bit is 0.

For more information about the common events and the use of the PMCEIDn registers see The section describing 'Event numbers and common events' in chapter D5 'The Performance Monitors Extension' of the ARM Architecture Reference Manual, for ARMv8-A architecture profile.

This register is part of the Performance Monitors registers functional group.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EPMAD	SLK	Default
Error	Error	Error	Error	RO	RO

Configuration

External register PMCEID3 is architecturally mapped to AArch64 System register [PMCEID1_EL0\[63:32\]](#).

External register PMCEID3 bits [63:32] are architecturally mapped to AArch32 System register [PMCEID3](#).

PMCEID3 is in the Core power domain.

This register is introduced in ARMv8.1.

Attributes

PMCEID3 is a 32-bit register.

Field descriptions

The PMCEID3 bit assignments are:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IDhi[63:32]																															

IDhi[63:32], bits [31:0]

IDhi[n] corresponds to common event (0x4000 + n).

For each bit:

IDhi[63:32]	Meaning
0	The common event is not implemented, or not counted.
1	The common event is implemented.

A bit that corresponds to a reserved event number is reserved. The value might be used in a future revision of the architecture to identify an additional common event.

Note

Such an event might be added retrospectively to an earlier version of the PMU architecture, provided the event does not require any additional PMU features and has an event number that can be represented in the PMCEID<n> registers of that earlier version of the PMU architecture.

Accessing the PMCEID3

PMCEID3 can be accessed through the external debug interface:

Component	Offset
PMU	0xE2C

28/09/2017 08:16:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

[SysReg_v83A_xml-00bet4](#)
(old)

htmldiff from-
SysReg_v83A_xml-00bet4

(new)
[SysReg_v83A_xml-00bet5](#)

PMPCSR, Program Counter Sample Register

The PMPCSR characteristics are:

Purpose

Holds a sampled instruction address value.

This register is part of the Performance Monitors registers functional group.

Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
Error	Error	Error	RO	RO

IMPLEMENTATION DEFINED extensions to external debug might make the value of this register UNKNOWN, see the section describing 'Permitted behavior that might make the PC Sample-based profiling registers UNKNOWN' in chapter H7 'The PC Sample-based Profiling Extension' of the ARM Architecture Reference Manual, for ARMv8-A architecture profile.

Configuration

PMPCSR is in the Core power domain.

Fields in this register reset to architecturally UNKNOWN values. These apply only on a Cold reset. The register is not affected by a Warm reset and is not affected by an External debug reset.

Implemented only when ARMv8.2-PCSample is implemented.

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.

This register is introduced in ARMv8.2.

Attributes

PMPCSR is a 64-bit register.

Field descriptions

The PMPCSR bit assignments are:

63	62	61	60	59	58	57	56	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	0	0	0	0	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]

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.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

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
00	Sample is from EL0.
01	Sample is from EL1.
10	Sample is from EL2.
11	Sample is from EL3.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

Bits [60:56]

Reserved, RES0.

PC Sample[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}.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

- 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 access has no side-effects.
- In any other cases, a read of PMPCSR[31:0] has the side-effect of indirectly writing to PMPCSR[63:32], PMCID1SR, PMCID2SR, and PMVIDSR:
 - If the PE is in Debug state, or PC Sample-based profiling is prohibited, PMPCSR[31:0] reads as 0xFFFFFFFF, and PMPCSR[63:32], PMCID1SR, PMCID2SR, and PMVIDSR become UNKNOWN.
 - If the PE is in Reset state, the sampled value is UNKNOWN and PMPCSR[63:32], PMCID1SR, PMCID2SR, and PMVIDSR become UNKNOWN.
 - If no instruction has been retired since the PE left Reset state, Debug state, or a state where PC Sample-based Profiling is prohibited, the sampled value is UNKNOWN, and PMPCSR[63:32], PMCID1SR, PMCID2SR, and PMVIDSR become UNKNOWN.

PC Sample[31:10], bits [31:10]

Bits[31:10] of the sampled instruction address value. The translation regime that PMPCSR samples can be determined from PMPCSR.{NS,EL}.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

SBZ, bit [0]

Reserved, SBZ.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

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 access has no side-effects.

In any other cases, a read of PMPCSR[31:0] has the side-effect of indirectly writing to PMPCSR[63:32], PMCID1SR, PMCID2SR, and PMVIDSR:

- If the PE is in Debug state, or PC Sample-based profiling is prohibited, PMPCSR[31:0] reads as 0xFFFFFFFF, and PMPCSR[63:32], PMCID1SR, PMCID2SR, and PMVIDSR become UNKNOWN.
- If the PE is in Reset state, the sampled value is UNKNOWN and PMPCSR[63:32], PMCID1SR, PMCID2SR, and PMVIDSR become UNKNOWN.

- If no instruction has been retired since the PE left Reset state, Debug state, or a state where PC Sample-based Profiling is prohibited, the sampled value is UNKNOWN, and PMPCSR.[63:32], [PMCID1SR](#), [PMCID2SR](#), and [PMVIDSR](#) become UNKNOWN.

Accessing the PMPCSR

PMPCSR[31:0] can be accessed through the external debug interface:

Component	Offset
PMU	0x200
PMU	0x220

PMPCSR[63:32] can be accessed through the external debug interface:

Component	Offset
PMU	0x204
PMU	0x224

28/0907/2017 0816:2440

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

SysReg_v83A_xml-00bet4 (old)	htmldiff from-	(new)
	SysReg_v83A_xml-00bet4	SysReg_v83A_xml-00bet5