

# 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 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 patents.

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, ARM makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, 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’s customers is not intended to create or refer to any partnership relationship with any other company. ARM may make changes to this document at any time and without notice.

If any of the provisions contained in these terms conflict with any of the provisions of any signed written agreement specifically covering this document with ARM, then the signed written agreement prevails over and supersedes the conflicting provisions of these terms.

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM Limited or its affiliates in the EU and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. You must follow the ARM trademark usage guidelines <http://www.arm.com/about/trademarks/guidelines/index.php>.

Copyright © 2017 ARM Limited or its affiliates. All rights reserved.

ARM Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

LES-PRE-20327

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

# 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

[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

[DBGDTRRXExt](#): Debug OS Lock Data Transfer Register, Receive, External View

[DBGDTRRXInt](#): Debug Data Transfer Register, Receive

[DBGDTRTXExt](#): Debug OS Lock Data Transfer Register, Transmit

[DBGDTRTXInt](#): Debug Data Transfer Register, Transmit

[DBGOSDLR](#): Debug OS Double Lock Register

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

[DBGOSLAR](#): Debug OS Lock Access Register

[DBGOSLSR](#): Debug OS Lock Status Register

[DBGPRCR](#): Debug Power Control Register

[DBGVCR](#): Debug Vector Catch Register

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

[DBGWFAR](#): Debug Watchpoint Fault Address Register

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

[DFAR](#): Data Fault Address Register

[DFSR](#): Data Fault Status Register

[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

[HAMAIRO](#): Hyp Auxiliary Memory Attribute Indirection Register 0

[HAMAIR1](#): Hyp Auxiliary Memory Attribute Indirection Register 1

[HCPTR](#): Hyp Architectural Feature Trap Register

[HCR](#): Hyp Configuration Register

[HCR2](#): Hyp Configuration Register 2

[HDCR](#): Hyp Debug Control Register

[HDFAR](#): Hyp Data Fault Address Register

[HIFAR](#): Hyp Instruction Fault Address Register

[HMAIRO](#): Hyp Memory Attribute Indirection Register 0

[HMAIR1](#): Hyp Memory Attribute Indirection Register 1

[HPFAR](#): Hyp IPA Fault Address Register

[HRMR](#): Hyp Reset Management Register

[HSCTLR](#): Hyp System Control Register

[HSR](#): Hyp Syndrome Register

[HSTR](#): Hyp System Trap Register

[HTCR](#): Hyp Translation Control Register

[HTPIDR](#): Hyp Software Thread ID Register

[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\\_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

[PMCCFILTER](#): 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

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

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

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

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

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

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

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

[PRRR](#): Primary Region Remap Register

[REVIDR](#): Revision ID Register

[RMR](#): Reset Management Register

[RVBAR](#): Reset Vector Base Address Register

[SCR](#): Secure Configuration Register

[SCTLR](#): System Control Register

[SDCR](#): Secure Debug Control Register

[SDER](#): Secure Debug Enable Register

[SPSR](#): Saved Program Status Register

[SPSR\\_abt](#): Saved Program Status Register (Abort mode)

[SPSR\\_fiq](#): Saved Program Status Register (FIQ mode)

[SPSR\\_hyp](#): Saved Program Status Register (Hyp mode)

[SPSR\\_irq](#): Saved Program Status Register (IRQ mode)

[SPSR\\_mon](#): Saved Program Status Register (Monitor mode)

[SPSR\\_svc](#): Saved Program Status Register (Supervisor mode)

[SPSR\\_und](#): Saved Program Status Register (Undefined mode)

[TCMTR](#): TCM Type Register

[TLBTR](#): TLB Type Register

[TPIDRPRW](#): PL1 Software Thread ID Register

[TPIDRURO](#): PL0 Read-Only Software Thread ID Register

[TPIDRURW](#): PL0 Read/Write Software Thread ID Register

[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

02/05/2017 15:43

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

# ACTLR, Auxiliary Control Register

The ACTLR characteristics are:

## Purpose

Provides IMPLEMENTATION DEFINED configuration and control options for execution at EL1 and EL0.

This register is part of:

- The Other system control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch32 System register ACTLR is architecturally mapped to AArch64 System register [ACTLR\\_EL1\[31:0\]](#).

Some bits might define global configuration settings, and be common to the Secure and Non-secure instances of the register.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

ACTLR is a 32-bit register.

## Field descriptions

The ACTLR 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the ACTLR

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, 1	000	001	0001	1111	0000

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	

EL3 using AArch32	x	x	0	-	n/a	n/a	RW	ACTLR_s
EL3 using AArch32	x	0	1	-	RW	RW	RW	ACTLR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	ACTLR_ns
EL3 not implemented	x	x	0	-	RW	n/a	n/a	ACTLR
EL3 not implemented	x	0	1	-	RW	RW	n/a	ACTLR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	ACTLR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	ACTLR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	ACTLR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	ACTLR

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](#).TACR==1, Non-secure accesses to this register from EL1 are trapped to EL2 using AArch64.
- 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](#).TACR==1, Non-secure accesses to this register from EL1 are trapped to EL2 using AArch64.
- 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](#).TAC==1, Non-secure 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.

# ACTLR2, Auxiliary Control Register 2

The ACTLR2 characteristics are:

## Purpose

Provides additional space to the ACTLR register to hold IMPLEMENTATION DEFINED trap functionality for execution at EL1 and EL0.

This register is part of:

- The Other system control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch32 System register ACTLR2 is architecturally mapped to AArch64 System register [ACTLR\\_EL1\[63:32\]](#).

In ARMv8.0 and ARMv8.1, it is IMPLEMENTATION DEFINED whether this register is implemented, or whether it causes UNDEFINED exceptions when accessed. The implementation of this register can be detected by examining [ID\\_MMFR4.AC2](#).

From ARMv8.2 this register must be implemented.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

ACTLR2 is a 32-bit register.

## Field descriptions

The ACTLR2 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the ACTLR2

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, 3	000	011	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	ACTLR2
EL3 not implemented	x	0	1	-	RW	RW	n/a	ACTLR2
EL3 not implemented	x	1	1	-	n/a	RW	n/a	ACTLR2
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	ACTLR2
EL3 using AArch64	x	0	1	-	RW	RW	n/a	ACTLR2
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	ACTLR2
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	ACTLR2_s
EL3 using AArch32	x	0	1	-	RW	RW	RW	ACTLR2_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	ACTLR2_ns

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](#).TACR==1, Non-secure accesses to this register from EL1 are trapped to EL2 using AArch64.
- 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](#).TACR==1, Non-secure accesses to this register from EL1 are trapped to EL2 using AArch64.
- 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](#).TAC==1, Non-secure 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.



# ADFSR, Auxiliary Data Fault Status Register

The ADFSR characteristics are:

## Purpose

Provides additional IMPLEMENTATION DEFINED fault status information for Data Abort exceptions taken to EL1 modes, and EL3 modes when EL3 is implemented and is using AArch32.

This register is part of:

- The Exception and fault handling registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch32 System register ADFSR is architecturally mapped to AArch64 System register [AFSR0\\_EL1](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

ADFSR is a 32-bit register.

## Field descriptions

The ADFSR 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the ADFSR

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>, c5, c1, 0	000	000	0101	1111	0001

## 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	ADFSR
EL3 not implemented	x	0	1	-	RW	RW	n/a	ADFSR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	ADFSR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	ADFSR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	ADFSR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	ADFSR
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	ADFSR_s
EL3 using AArch32	x	0	1	-	RW	RW	RW	ADFSR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	ADFSR_ns

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<sup>®</sup> 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.T5](#)==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.T5](#)==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.T5](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# AIDR, Auxiliary ID Register

The AIDR characteristics are:

## Purpose

Provides IMPLEMENTATION DEFINED identification information.

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

This register is part of:

- The Identification registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

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

AArch32 System register AIDR is architecturally mapped to AArch64 System register [AIDR\\_EL1](#).

## Attributes

AIDR is a 32-bit register.

## Field descriptions

The AIDR 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the AIDR

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, 1, <Rt>, c0, c0, 7	001	111	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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HCR\\_EL2.TID1](#)=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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- If [HCR\\_EL2.TID1](#)=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  $\text{SCR\_EL3.NS}=1$  :

- If [HCR.TID1](#)=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.

02/05/2017 15:43

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

# AIFSR, Auxiliary Instruction Fault Status Register

The AIFSR characteristics are:

## Purpose

Provides additional IMPLEMENTATION DEFINED fault status information for Prefetch Abort exceptions taken to EL1 modes, and EL3 modes when EL3 is implemented and is using AArch32.

This register is part of:

- The Exception and fault handling registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch32 System register AIFSR is architecturally mapped to AArch64 System register [AFSR1\\_EL1](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

AIFSR is a 32-bit register.

## Field descriptions

The AIFSR 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the AIFSR

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>, c5, c1, 1	000	001	0101	1111	0001

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	

EL3 using AArch32	x	x	0	-	n/a	n/a	RW	AIFSR_s
EL3 not implemented	x	x	0	-	RW	n/a	n/a	AIFSR
EL3 not implemented	x	0	1	-	RW	RW	n/a	AIFSR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	AIFSR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	AIFSR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	AIFSR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	AIFSR
EL3 using AArch32	x	0	1	-	RW	RW	RW	AIFSR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	AIFSR_ns

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.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.T5](#)==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.T5](#)==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.T5](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# AMAIR0, Auxiliary Memory Attribute Indirection Register 0

The AMAIR0 characteristics are:

## Purpose

When using the Long-descriptor format translation tables for stage 1 translations, provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by [MAIR0](#).

This register is part of:

- The Virtual memory control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch32 System register AMAIR0 is architecturally mapped to AArch64 System register [AMAIR\\_EL1\[31:0\]](#).

When EL3 is using AArch32, write access to AMAIR0(S) is disabled when the CP15SDISABLE signal is asserted HIGH.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

AMAIR0 is a 32-bit register.

## Field descriptions

The AMAIR0 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
IMPLEMENTATION DEFINED																															

This register is RES0 in the following cases:

- When an implementation does not provide any IMPLEMENTATION DEFINED memory attributes.
- When the Long-descriptor translation table format is not used.

If EL3 is implemented and is using AArch32:

- AMAIR0(S) gives the value for memory accesses from Secure state.
- AMAIR0(NS) gives the value for memory accesses from Non-secure states other than Hyp mode.

Any IMPLEMENTATION DEFINED memory attributes are additional qualifiers for the memory locations and must not change the architected behavior specified by [MAIR0](#) and [MAIR1](#).

In a typical implementation, AMAIR0 and [AMAIR1](#) split into eight one-byte fields, corresponding to the MAIRn.Attr<n> fields, but the architecture does not require them to do so.

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the AMAIR0

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 &lt;syntax&gt;

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

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c10, c3, 0	000	000	1010	1111	0011

## 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	AMAIRO
EL3 not implemented	x	0	1	-	RW	RW	n/a	AMAIRO
EL3 not implemented	x	1	1	-	n/a	RW	n/a	AMAIRO
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	AMAIRO
EL3 using AArch64	x	0	1	-	RW	RW	n/a	AMAIRO
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	AMAIRO
EL3 using AArch32	x	0	1	-	RW	RW	RW	AMAIRO_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	AMAIRO_ns
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	AMAIRO_s

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

When EL3 is using AArch32, write access to AMAIRO\_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.T10==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.T10==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.T10==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.



# AMAIR1, Auxiliary Memory Attribute Indirection Register 1

The AMAIR1 characteristics are:

## Purpose

When using the Long-descriptor format translation tables for stage 1 translations, provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by [MAIR1](#).

This register is part of:

- The Virtual memory control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch32 System register AMAIR1 is architecturally mapped to AArch64 System register [AMAIR\\_EL1\[63:32\]](#).

When EL3 is using AArch32, write access to AMAIR1(S) is disabled when the CP15SDISABLE signal is asserted HIGH.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

AMAIR1 is a 32-bit register.

## Field descriptions

The AMAIR1 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
IMPLEMENTATION DEFINED																															

This register is RES0 in the following cases:

- When an implementation does not provide any IMPLEMENTATION DEFINED memory attributes.
- When the Long-descriptor translation table format is not used.

If EL3 is implemented and is using AArch32:

- AMAIR1(S) gives the value for memory accesses from Secure state.
- AMAIR1(NS) gives the value for memory accesses from Non-secure states other than Hyp mode.

Any IMPLEMENTATION DEFINED memory attributes are additional qualifiers for the memory locations and must not change the architected behavior specified by [MAIR0](#) and [MAIR1](#).

In a typical implementation, [AMAIR0](#) and AMAIR1 split into eight one-byte fields, corresponding to the MAIRn.Attr<n> fields, but the architecture does not require them to do so.

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the AMAIR1

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 &lt;syntax&gt;

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

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c10, c3, 1	000	001	1010	1111	0011

## 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	AMAIR1
EL3 not implemented	x	0	1	-	RW	RW	n/a	AMAIR1
EL3 not implemented	x	1	1	-	n/a	RW	n/a	AMAIR1
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	AMAIR1
EL3 using AArch64	x	0	1	-	RW	RW	n/a	AMAIR1
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	AMAIR1
EL3 using AArch32	x	0	1	-	RW	RW	RW	AMAIR1_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	AMAIR1_ns
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	AMAIR1_s

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

When EL3 is using AArch32, write access to AMAIR1\_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.T10==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.T10==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.T10==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.

# APSR, Application Program Status Register

The APSR characteristics are:

## Purpose

Hold program status and control information.

This register is part of the Process state registers functional group.

## Usage constraints

The APSR can be read using the MRS instruction and written using the MSR (immediate) or MSR (register) instructions. For more details on the instruction syntax, see 'PSTATE and banked register access instructions' in the ARMv8 ARM, section F1.5.

## Traps and Enables

There are no traps or enables affecting this register.

## Configuration

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

## Attributes

APSR is a 32-bit register.

## Field descriptions

The APSR 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
N	Z	C	V	Q	0	0	0	0	0	0	0	GE			0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

### N, bit [31]

Negative condition flag. Set to bit[31] of the result of the last flag-setting instruction. If the result is regarded as a two's complement signed integer, then N is set to 1 if the result was negative, and N is set to 0 if the result was positive or zero.

### Z, bit [30]

Zero condition flag. Set to 1 if the result of the last flag-setting instruction was zero, and to 0 otherwise. A result of zero often indicates an equal result from a comparison.

### C, bit [29]

Carry condition flag. Set to 1 if the last flag-setting instruction resulted in a carry condition, for example an unsigned overflow on an addition.

### V, bit [28]

Overflow condition flag. Set to 1 if the last flag-setting instruction resulted in an overflow condition, for example a signed overflow on an addition.

**Q, bit [27]**

Cumulative saturation bit. Set to 1 to indicate that overflow or saturation occurred in some instructions.

**Bits [26:20]**

Reserved, RES0.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**Bits [15:5]**

Reserved, RES0.

**Bit [4]**

Reserved, RES1.

**Bits [3:0]**

Reserved, RES0.

02/05/2017 15:43

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

# CCSIDR, Current Cache Size ID Register

The CCSIDR characteristics are:

## Purpose

Provides information about the architecture of the currently selected cache.

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 CCSIDR is architecturally mapped to AArch64 System register [CCSIDR\\_EL1](#).

The implementation includes one CCSIDR for each cache that it can access. [CSSELR](#) and the Security state select which Cache Size ID Register is accessible.

## Attributes

CCSIDR is a 32-bit register.

## Field descriptions

The CCSIDR 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
UNKNOWN				NumSets																Associativity								LineSize			

### UNKNOWN, bits [31:28]

Reserved, UNKNOWN.

### NumSets, bits [27:13]

(Number of sets in cache) - 1, therefore a value of 0 indicates 1 set in the cache. The number of sets does not have to be a power of 2.

### Associativity, bits [12:3]

(Associativity of cache) - 1, therefore a value of 0 indicates an associativity of 1. The associativity does not have to be a power of 2.

### LineSize, bits [2:0]

( $\log_2(\text{Number of bytes in cache line})$ ) - 4. For example:

For a line length of 16 bytes:  $\log_2(16) = 4$ , LineSize entry = 0. This is the minimum line length.

For a line length of 32 bytes:  $\log_2(32) = 5$ , LineSize entry = 1.

---

### Note

The parameters NumSets, Associativity, and LineSize in these registers define the architecturally visible parameters that are required for the cache maintenance by Set/Way instructions. They are not guaranteed to represent the actual microarchitectural features of a

---

design. You cannot make any inference about the actual sizes of caches based on these parameters.

## Accessing the CCSIDR

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, 1, <Rt>, c0, c0, 0	001	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.

If [CSSELR](#).Level is programmed to a cache level that is not implemented, then on a read of the CCSIDR the behavior is CONSTRAINED UNPREDICTABLE, and can be one of the following:

- The CCSIDR read is treated as NOP.
- The CCSIDR read is UNDEFINED.
- The CCSIDR read returns an UNKNOWN value.

## 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](#).TID2==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](#).TID2==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](#).TID2==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.



# CLIDR, Cache Level ID Register

The CLIDR characteristics are:

## Purpose

Identifies the type of cache, or caches, that are implemented at each level and can be managed using the architected cache maintenance instructions that operate by set/way, up to a maximum of seven levels. Also identifies the Level of Coherence (LoC) and Level of Unification (LoU) for the cache hierarchy.

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 CLIDR is architecturally mapped to AArch64 System register [CLIDR\\_EL1](#).

## Attributes

CLIDR is a 32-bit register.

## Field descriptions

The CLIDR 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
ICB	LoUU	LoC	LoUIS	Ctype7	Ctype6	Ctype5	Ctype4	Ctype3	Ctype2	Ctype1																					

### ICB, bits [31:30]

Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions.

The possible values are:

ICB	Meaning
00	Not disclosed by this mechanism.
01	L1 cache is the highest Inner Cacheable level.
10	L2 cache is the highest Inner Cacheable level.
11	L3 cache is the highest Inner Cacheable level.

### LoUU, bits [29:27]

Level of Unification Uniprocessor for the cache hierarchy.

### LoC, bits [26:24]

Level of Coherence for the cache hierarchy.

### LoUIS, bits [23:21]

Level of Unification Inner Shareable for the cache hierarchy.



**Ctype<n>, bits [3(n-1)+2:3(n-1)], for n = 1 to 7**

Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are:

Ctype<n>	Meaning
000	No cache.
001	Instruction cache only.
010	Data cache only.
011	Separate instruction and data caches.
100	Unified cache.

All other values are reserved.

If software reads the Cache Type fields from Ctype1 upwards, once it has seen a value of 000, no caches that can be managed using the architected cache maintenance instructions that operate by set/way exist at further-out levels of the hierarchy. So, for example, if Ctype3 is the first Cache Type field with a value of 000, the values of Ctype4 to Ctype7 must be ignored.

## Accessing the CLIDR

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, 1, <Rt>, c0, c0, 1	001	001	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<sup>®</sup> 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.TID2](#)==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.TID2](#)==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.TID2](#)==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.

02/05/2017 15:43

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

# CNTFRQ, Counter-timer Frequency register

The CNTFRQ characteristics are:

## Purpose

This register is provided so that software can discover the frequency of the system counter. It must be programmed with this value as part of system initialization. The value of the register is not interpreted by hardware.

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 CNTFRQ is architecturally mapped to AArch64 System register [CNTFRQ\\_EL0](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTFRQ is a 32-bit register.

## Field descriptions

The CNTFRQ 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
Clock frequency																															

### Bits [31:0]

Clock frequency. Indicates the system counter clock frequency, in Hz.

## Accessing the CNTFRQ

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, c0, 0	000	000	1110	1111	0000

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
EL1 is the highest implemented Exception level	x	x	x	RO	RW	n/a	n/a

EL2 is the highest implemented Exception level	x	0	1	RO	RO	RW	n/a
EL2 is the highest implemented Exception level	x	1	1	RO	n/a	RW	n/a
EL3 is the highest implemented Exception level	x	x	0	RO	RO	RO	RW
EL3 is the highest implemented Exception level	x	0	1	RO	RO	RO	RW
EL3 is the highest implemented Exception level	x	1	1	RO	n/a	RO	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 `HCR_EL2.E2H==0` :

- If `CNTKCTL_EL1.EL0PCTEN==0`, and `CNTKCTL_EL1.EL0VCTEN==0`, read accesses to this register from EL0 are trapped to EL1.
- If `CNTKCTL.PL0PCTEN==0`, and `CNTKCTL.PL0VCTEN==0`, read 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.EL0PCTEN==0`, and `CNTKCTL_EL1.EL0VCTEN==0`, Non-secure read accesses to this register from EL0 are trapped to EL1.
- If `CNTKCTL.PL0PCTEN==0`, and `CNTKCTL.PL0VCTEN==0`, Non-secure read 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.EL0PCTEN==0`, and `CNTHCTL_EL2.EL0VCTEN==0`, Non-secure read accesses to this register from EL0 are trapped to EL2.

02/05/2017 15:43

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

# CNTHCTL, Counter-timer Hyp Control register

The CNTHCTL characteristics are:

## Purpose

Controls the generation of an event stream from the physical counter, and access from Non-secure EL1 modes to the physical counter and the Non-secure EL1 physical timer.

This register is part of:

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

## Configuration

AArch32 System register CNTHCTL is architecturally mapped to AArch64 System register [CNTHCTL\\_EL2](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTHCTL is a 32-bit register.

## Field descriptions

The CNTHCTL 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	EVNTI	EVNTDIR	EVNTEN	PL1PCEN	PL1PCTEN			

### Bits [31:8]

Reserved, RES0.

### EVNTI, bits [7:4]

Selects which bit (0 to 15) of the counter register [CNTPCT](#) is the trigger for the event stream generated from that counter, when that stream is enabled.

### EVNTDIR, bit [3]

Controls which transition of the counter register [CNTPCT](#) trigger bit, defined by EVNTI, generates an event when the event stream is enabled:

EVNTDIR	Meaning
0	A 0 to 1 transition of the trigger bit triggers an event.
1	A 1 to 0 transition of the trigger bit triggers an event.

### EVNTEN, bit [2]

Enables the generation of an event stream from the counter register [CNTPCT](#):

EVNTEN	Meaning
0	Disables the event stream.
1	Enables the event stream.

**PL1PCEN, bit [1]**

Traps Non-secure EL0 and EL1 accesses to the physical timer registers to Hyp mode.

PL1PCEN	Meaning
0	Non-secure EL0 and EL1 accesses to the <a href="#">CNTP_CTL</a> , <a href="#">CNTP_CVAL</a> , and <a href="#">CNTP_TVAL</a> are trapped to Hyp mode, unless the it is trapped by <a href="#">CNTKCTL.PL0PTEN</a> .
1	This control does not cause any instructions to be trapped.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 1 other than for the purpose of a direct read.

**PL1PCTEN, bit [0]**

Traps Non-secure EL0 and EL1 accesses to the physical counter register to Hyp mode.

PL1PCTEN	Meaning
0	Non-secure EL0 and EL1 accesses to the <a href="#">CNTPCT</a> are trapped to Hyp mode, unless it is trapped by <a href="#">CNTKCTL.PL0PCTEN</a> .
1	This control does not cause any instructions to be trapped.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 1 other than for the purpose of a direct read.

**Accessing the CNTHCTL**

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, c1, 0	100	000	1110	1111	0001

**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.

# CNTHP\_CTL, Counter-timer Hyp Physical Timer Control register

The CNTHP\_CTL characteristics are:

## Purpose

Control register for the Hyp mode physical timer.

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

## Configuration

AArch32 System register CNTHP\_CTL is architecturally mapped to AArch64 System register [CNTHP\\_CTL\\_EL2](#).

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 with EL2 using AArch32, or into EL3 with EL3 using AArch32. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTHP\_CTL is a 32-bit register.

## Field descriptions

The CNTHP\_CTL 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	<a href="#">ISTATUS</a>	<a href="#">IMASK</a>	<a href="#">ENABLE</a>

### Bits [31:3]

Reserved, RES0.

### ISTATUS, bit [2]

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0	Timer condition is not met.
1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.

For more information see 'Operation of the CompareValue views of the timers' and 'Operation of the TimerValue views of the timers' in the ARM ARM, chapter D6.

This bit is read-only.

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

### IMASK, bit [1]

Timer interrupt mask bit. Permitted values are:

IMASK	Meaning
0	Timer interrupt is not masked by the IMASK bit.
1	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

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

## ENABLE, bit [0]

Enables the timer. Permitted values are:

ENABLE	Meaning
0	Timer disabled.
1	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from [CNTHP\\_TVAL](#) continues to count down.

### Note

Disabling the output signal might be a power-saving option.

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

## Accessing the CNTHP\_CTL

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, 1	100	001	1110	1111	0010
p15, 0, <Rt>, c14, c2, 1	000	001	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, 1	x	x	0	-	-	n/a	-
p15, 4, <Rt>, c14, c2, 1	x	0	1	-	-	RW	RW
p15, 4, <Rt>, c14, c2, 1	x	1	1	-	n/a	RW	RW
p15, 0, <Rt>, c14, c2, 1	x	x	0	<a href="#">CNTP_CTL</a>	<a href="#">CNTP_CTL</a>	n/a	<a href="#">CNTP_CTL</a>
p15, 0, <Rt>, c14, c2, 1	0	0	1	<a href="#">CNTP_CTL</a>	<a href="#">CNTP_CTL</a>	<a href="#">CNTP_CTL</a>	<a href="#">CNTP_CTL</a>
p15, 0, <Rt>, c14, c2, 1	0	1	1	<a href="#">CNTP_CTL</a>	n/a	<a href="#">CNTP_CTL</a>	<a href="#">CNTP_CTL</a>
p15, 0, <Rt>, c14, c2, 1	1	0	1	<a href="#">CNTP_CTL</a>	<a href="#">CNTP_CTL</a>	n/a	n/a
p15, 0, <Rt>, c14, c2, 1	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<sup>®</sup> 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.

02/05/2017 15:43

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

# 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																															
31	30	29	28	27	26	25	24	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.ENABLE](#) is 1, the timer condition is met when ([CNTPCT](#) - CompareValue) is greater than 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	<a href="#">CNTP_CVAL</a>	<a href="#">CNTP_CVAL</a>	n/a	<a href="#">CNTP_CVAL</a>
p15, 2, <Rt>, <Rt2>, c14	0	0	1	<a href="#">CNTP_CVAL</a>	<a href="#">CNTP_CVAL</a>	<a href="#">CNTP_CVAL</a>	<a href="#">CNTP_CVAL</a>
p15, 2, <Rt>, <Rt2>, c14	0	1	1	<a href="#">CNTP_CVAL</a>	n/a	<a href="#">CNTP_CVAL</a>	<a href="#">CNTP_CVAL</a>
p15, 2, <Rt>, <Rt2>, c14	1	0	1	<a href="#">CNTP_CVAL</a>	<a href="#">CNTP_CVAL</a>	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<sup>®</sup> 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.

02/05/2017 15:43

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

# 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  $(\text{CNTHP\_CVAL} - \text{CNTPCT})$ .

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

When [CNTHP\\_CTL.ENABLE](#) is 1, the timer condition is met when  $(\text{CNTPCT} - \text{CNTHP\_CVAL})$  is greater than 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	<a href="#">CNTP_TVAL</a>	<a href="#">CNTP_TVAL</a>	n/a	<a href="#">CNTP_TVAL</a>
p15, 0, <Rt>, c14, c2, 0	0	0	1	<a href="#">CNTP_TVAL</a>	<a href="#">CNTP_TVAL</a>	<a href="#">CNTP_TVAL</a>	<a href="#">CNTP_TVAL</a>
p15, 0, <Rt>, c14, c2, 0	0	1	1	<a href="#">CNTP_TVAL</a>	n/a	<a href="#">CNTP_TVAL</a>	<a href="#">CNTP_TVAL</a>
p15, 0, <Rt>, c14, c2, 0	1	0	1	<a href="#">CNTP_TVAL</a>	<a href="#">CNTP_TVAL</a>	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<sup>®</sup> 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.

02/05/2017 15:43

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

# CNTHV\_CTL, Counter-timer Virtual Timer Control register (EL2)

The CNTHV\_CTL characteristics are:

## Purpose

Provides AArch32 access to the control register 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\_CTL is architecturally mapped to AArch64 System register [CNTHV\\_CTL\\_EL2](#).

This register is introduced in ARMv8.1.

## Attributes

CNTHV\_CTL is a 32-bit register.

## Field descriptions

The CNTHV\_CTL 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	ISTATUS	IMASK	ENABLE

### Bits [31:3]

Reserved, RES0.

### ISTATUS, bit [2]

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0	Timer condition is not met.
1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.

For more information see 'Operation of the CompareValue views of the timers' and 'Operation of the TimerValue views of the timers' in the ARM ARM, chapter D6.

This bit is read-only.

**IMASK, bit [1]**

Timer interrupt mask bit. Permitted values are:

IMASK	Meaning
0	Timer interrupt is not masked by the IMASK bit.
1	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

**ENABLE, bit [0]**

Enables the timer. Permitted values are:

ENABLE	Meaning
0	Timer disabled.
1	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from [CNTHV\\_TVAL](#) continues to count down.

**Note**

Disabling the output signal might be a power-saving option.

**Accessing the CNTHV\_CTL**

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, 1	000	001	1110	1111	0011

This register is accessed using the encoding for [CNTV\\_CTL](#).

**Accessibility**

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	<a href="#">CNTV_CTL</a>	<a href="#">CNTV_CTL</a>	n/a	<a href="#">CNTV_CTL</a>
0	0	1	<a href="#">CNTV_CTL</a>	<a href="#">CNTV_CTL</a>	<a href="#">CNTV_CTL</a>	<a href="#">CNTV_CTL</a>
0	1	1	<a href="#">CNTV_CTL</a>	n/a	<a href="#">CNTV_CTL</a>	<a href="#">CNTV_CTL</a>
1	0	1	<a href="#">CNTV_CTL</a>	<a href="#">CNTV_CTL</a>	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<sup>®</sup> 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.

02/05/2017 15:43

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



# 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 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 &lt;syntax&gt;

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	<a href="#">CNTV_CVAL</a>	<a href="#">CNTV_CVAL</a>	n/a	<a href="#">CNTV_CVAL</a>
0	0	1	<a href="#">CNTV_CVAL</a>	<a href="#">CNTV_CVAL</a>	<a href="#">CNTV_CVAL</a>	<a href="#">CNTV_CVAL</a>
0	1	1	<a href="#">CNTV_CVAL</a>	n/a	<a href="#">CNTV_CVAL</a>	<a href="#">CNTV_CVAL</a>
1	0	1	<a href="#">CNTV_CVAL</a>	<a href="#">CNTV_CVAL</a>	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<sup>®</sup> 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.

02/05/2017 15:43

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

# 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 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	<a href="#">CNTV_TVAL</a>	<a href="#">CNTV_TVAL</a>	n/a	<a href="#">CNTV_TVAL</a>
0	0	1	<a href="#">CNTV_TVAL</a>	<a href="#">CNTV_TVAL</a>	<a href="#">CNTV_TVAL</a>	<a href="#">CNTV_TVAL</a>
0	1	1	<a href="#">CNTV_TVAL</a>	n/a	<a href="#">CNTV_TVAL</a>	<a href="#">CNTV_TVAL</a>
1	0	1	<a href="#">CNTV_TVAL</a>	<a href="#">CNTV_TVAL</a>	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<sup>®</sup> 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.

# CNTKCTL, Counter-timer Kernel Control register

The CNTKCTL characteristics are:

## Purpose

Controls the generation of an event stream from the virtual counter, and access from EL0 modes to the physical counter, virtual counter, EL1 physical timers, and the virtual timer.

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 CNTKCTL is architecturally mapped to AArch64 System register [CNTKCTL\\_ELI](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTKCTL is a 32-bit register.

## Field descriptions

The CNTKCTL 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	PL0PTEN	PL0VTEN	EVNTI	EVNTDIR	EVNTEN	PL0VCTEN	PL0PCTEN		

### Bits [31:10]

Reserved, RES0.

### PL0PTEN, bit [9]

Traps PL0 accesses to the physical timer registers to Undefined mode.

PL0PTEN	Meaning
0	PL0 accesses to the <a href="#">CNTP_CTL</a> , <a href="#">CNTP_CVAL</a> , and <a href="#">CNTP_TVAL</a> registers are trapped to Undefined mode.
1	This control does not cause any instructions to be trapped.

### PL0VTEN, bit [8]

Traps PL0 accesses to the virtual timer registers to Undefined mode.

PL0VTEN	Meaning
0	PL0 accesses to the <a href="#">CNTV_CTL</a> , <a href="#">CNTV_CVAL</a> , and <a href="#">CNTV_TVAL</a> registers are trapped to Undefined mode.
1	This control does not cause any instructions to be trapped.

### EVNTI, bits [7:4]

Selects which bit (0 to 15) of the counter register [CNTVCT](#) is the trigger for the event stream generated from that counter, when that stream is enabled.

**EVNTDIR, bit [3]**

Controls which transition of the counter register [CNTVCT](#) trigger bit, defined by EVNTI, generates an event when the event stream is enabled:

EVNTDIR	Meaning
0	A 0 to 1 transition of the trigger bit triggers an event.
1	A 1 to 0 transition of the trigger bit triggers an event.

**EVNTEN, bit [2]**

Enables the generation of an event stream from the counter register [CNTVCT](#):

EVNTEN	Meaning
0	Disables the event stream.
1	Enables the event stream.

**PL0VCTEN, bit [1]**

Traps PL0 accesses to the frequency register and virtual counter register to Undefined mode.

PL0VCTEN	Meaning
0	PL0 accesses to the <a href="#">CNTVCT</a> are trapped to Undefined mode. PL0 accesses to the <a href="#">CNTFRQ</a> register are trapped to Undefined mode, if <a href="#">CNTKCTL</a> .PL0PCTEN is also 0.
1	This control does not cause any instructions to be trapped.

**PL0PCTEN, bit [0]**

Traps PL0 accesses to the frequency register and physical counter register to Undefined mode.

PL0PCTEN	Meaning
0	PL0 accesses to the <a href="#">CNTPCT</a> are trapped to Undefined mode. PL0 accesses to the <a href="#">CNTFRQ</a> register are trapped to Undefined mode, if <a href="#">CNTKCTL</a> .PL0VCTEN is also 0.
1	This control does not cause any instructions to be trapped.

**Accessing the CNTKCTL**

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, c1, 0	000	000	1110	1111	0001

**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.

02/05/2017 15:43

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

# CNTPCT, Counter-timer Physical Count register

The CNTPCT characteristics are:

## Purpose

Holds the 64-bit physical count value.

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 CNTPCT is architecturally mapped to AArch64 System register [CNTPCT\\_EL0](#).

## Attributes

CNTPCT is a 64-bit register.

## Field descriptions

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

### Bits [63:0]

Physical count value.

## Accessing the CNTPCT

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

MRRC <syntax>

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

<syntax>	opc1	coproc	CRm
p15, 0, <Rt>, <Rt2>, c14	0000	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.

When HCR\_EL2.E2H==0 :

- If [CNTKCTL\\_EL1](#).EL0PCTEN==0, read accesses to this register from EL0 are trapped to EL1.
- If [CNTKCTL](#).PL0PCTEN==0, read 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](#).EL1PCTEN==0, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [CNTHCTL\\_EL2](#).EL1PCTEN==0, and [CNTKCTL\\_EL1](#).EL0PCTEN==1, Non-secure read 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](#).EL1PCTEN==0, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [CNTHCTL\\_EL2](#).EL1PCTEN==0, and [CNTKCTL\\_EL1](#).EL0PCTEN==1, Non-secure read accesses to this register from EL0 are trapped to EL2.
- If [CNTKCTL\\_EL1](#).EL0PCTEN==0, Non-secure read accesses to this register from EL0 are trapped to EL1.
- If [CNTKCTL](#).PL0PCTEN==0, Non-secure read 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](#).EL0PCTEN==0, Non-secure read 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](#).PL1PCTEN==0, Non-secure read accesses to this register from EL0 and EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# CNTP\_CTL, Counter-timer Physical Timer Control register

The CNTP\_CTL characteristics are:

## Purpose

Control register for the EL1 physical timer.

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

## Configuration

AArch32 System register CNTP\_CTL is architecturally mapped to AArch64 System register [CNTP\\_CTL\\_EL0](#).

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

CNTP\_CTL is a 32-bit register.

## Field descriptions

The CNTP\_CTL 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	ISTATUS	IMASK	ENABLE

### Bits [31:3]

Reserved, RES0.

### ISTATUS, bit [2]

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0	Timer condition is not met.
1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.

For more information see 'Operation of the CompareValue views of the timers' and 'Operation of the TimerValue views of the timers' in the ARM ARM, chapter D6.

This bit is read-only.

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

### IMASK, bit [1]

Timer interrupt mask bit. Permitted values are:

IMASK	Meaning
0	Timer interrupt is not masked by the IMASK bit.
1	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

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

## ENABLE, bit [0]

Enables the timer. Permitted values are:

ENABLE	Meaning
0	Timer disabled.
1	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from [CNTP\\_TVAL](#) continues to count down.

### Note

Disabling the output signal might be a power-saving option.

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

## Accessing the CNTP\_CTL

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, 1	000	001	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_CTL
EL3 not implemented	0	0	1	RW	RW	RW	n/a	CNTP_CTL
EL3 not implemented	0	1	1	RW	n/a	RW	n/a	CNTP_CTL
EL3 not implemented	1	0	1	RW	RW	n/a	n/a	CNTP_CTL
EL3 not implemented	1	1	1	<a href="#">CNTHP_CTL</a>	n/a	n/a	n/a	-
EL3 using AArch64	x	x	0	RW	RW	n/a	n/a	CNTP_CTL
EL3 using AArch64	0	0	1	RW	RW	RW	n/a	CNTP_CTL
EL3 using AArch64	0	1	1	RW	n/a	RW	n/a	CNTP_CTL
EL3 using AArch64	1	0	1	RW	RW	n/a	n/a	CNTP_CTL
EL3 using AArch64	1	1	1	<a href="#">CNTHP_CTL</a>	n/a	n/a	n/a	-
EL3 using AArch32	x	x	0	RW	n/a	n/a	RW	CNTP_CTL_s
EL3 using AArch32	x	0	1	RW	RW	RW	RW	CNTP_CTL_ns
EL3 using AArch32	x	1	1	RW	n/a	RW	RW	CNTP_CTL_ns

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.

02/05/2017 15:43

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

# 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																															
31	30	29	28	27	26	25	24	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 EL1 physical timer CompareValue.

When [CNTP\\_CTL](#).ENABLE is 1, the timer condition is met when ([CNTPCT](#) - CompareValue) is greater than 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	<a href="#">CNTHP_CVAL</a>	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	<a href="#">CNTHP_CVAL</a>	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](#).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.



# 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\\_ELO](#).

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 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	<a href="#">CNTHP_TVAL</a>	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	<a href="#">CNTHP_TVAL</a>	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](#).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.

02/05/2017 15:43

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

# CNTVCT, Counter-timer Virtual Count register

The CNTVCT characteristics are:

## Purpose

Holds the 64-bit virtual count value. The virtual count value is equal to the physical count value visible in [CNTPCT](#) minus the virtual offset visible in [CNTVOFF](#).

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 CNTVCT is architecturally mapped to AArch64 System register [CNTVCT\\_ELO](#).

The value of this register is the same as the value of [CNTPCT](#) in the following conditions:

- When EL2 is not implemented.
- When EL2 is implemented and is using AArch64, [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, and this register is read from Non-secure EL0.

## Attributes

CNTVCT is a 64-bit register.

## Field descriptions

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

### Bits [63:0]

Virtual count value.

## Accessing the CNTVCT

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

```
MRRC <syntax>
```

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

<syntax>	opc1	coproc	CRm
p15, 1, <Rt>, <Rt2>, c14	0001	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.

When HCR\_EL2.E2H==0 :

- If [CNTKCTL.PL0VCTEN](#)==0, read accesses to this register from EL0 are trapped to Undefined mode.
- If [CNTKCTL\\_EL1.EL0VCTEN](#)==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==1 && HCR\_EL2.TGE==0 :

- If [CNTKCTL.PL0VCTEN](#)==0, Non-secure read accesses to this register from EL0 are trapped to Undefined mode.
- If [CNTKCTL\\_EL1.EL0VCTEN](#)==0, Non-secure 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==1 && HCR\_EL2.TGE==1 :

- If [CNTHCTL\\_EL2.EL0PCTEN](#)==0, Non-secure read accesses to this register from EL0 are trapped to EL2.

02/05/2017 15:43

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

# CNTVOFF, Counter-timer Virtual Offset register

The CNTVOFF characteristics are:

## Purpose

Holds the 64-bit virtual offset. This is the offset between the physical count value visible in [CNTPCT](#) and the virtual count value visible in [CNTVCT](#).

This register is part of:

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

## Configuration

AArch32 System register CNTVOFF is architecturally mapped to AArch64 System register [CNTVOFF\\_EL2](#).

If EL2 is not implemented, this register is RES0 from EL3 and the virtual counter uses a fixed virtual offset of zero.

### Note

When EL2 is implemented and is using AArch64, if [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, the virtual counter uses a fixed virtual offset of zero when [CNTVCT](#) is read from Non-secure EL0.

When EL2 is implemented and can use AArch32, on a reset into an Exception level that is using AArch32 this register resets to an IMPLEMENTATION DEFINED value that might be UNKNOWN.

## Attributes

CNTVOFF is a 64-bit register.

## Field descriptions

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

### Bits [63:0]

Virtual offset.

## Accessing the CNTVOFF

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, 4, <Rt>, <Rt2>, c14	0100	1111	1110
--------------------------	------	------	------

## 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.

02/05/2017 15:43

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

# CNTV\_CTL, Counter-timer Virtual Timer Control register

The CNTV\_CTL characteristics are:

## Purpose

Control register 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\_CTL is architecturally mapped to AArch64 System register [CNTV\\_CTL\\_EL0](#).

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. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTV\_CTL is a 32-bit register.

## Field descriptions

The CNTV\_CTL 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	<a href="#">ISTATUS</a>	<a href="#">IMASK</a>	<a href="#">ENABLE</a>

### Bits [31:3]

Reserved, RES0.

### ISTATUS, bit [2]

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0	Timer condition is not met.
1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.

For more information see 'Operation of the CompareValue views of the timers' and 'Operation of the TimerValue views of the timers' in the ARM ARM, chapter D6.

This bit is read-only.

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

### IMASK, bit [1]

Timer interrupt mask bit. Permitted values are:

IMASK	Meaning
0	Timer interrupt is not masked by the IMASK bit.
1	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

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

## ENABLE, bit [0]

Enables the timer. Permitted values are:

ENABLE	Meaning
0	Timer disabled.
1	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from [CNTV\\_TVAL](#) continues to count down.

### Note

Disabling the output signal might be a power-saving option.

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

## Accessing the CNTV\_CTL

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, 1	000	001	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	<a href="#">CNTHV_CTL</a>	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 [CINTHCTL\\_EL2](#).EL0VTEN==0, Non-secure accesses to this register from EL0 are trapped to EL2.

02/05/2017 15:43

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

# 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																															

### 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 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	<a href="#">CNTHV_CVAL</a>	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.

02/05/2017 15:43

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

# 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\\_EL0](#).

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  $(\text{CNTV\_CVAL} - (\text{CNTPCT} - \text{CNTVOFF}))$ .

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

When [CNTP\\_CTL.ENABLE](#) is 1, the timer condition is met when  $(\text{CNTPCT} - \text{CNTP\_CVAL})$  is greater than 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	<a href="#">CNTHV_TVAL</a>	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<sup>®</sup> 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.

# CONTEXTIDR, Context ID Register

The CONTEXTIDR characteristics are:

## Purpose

Identifies the current Process Identifier and, when using the Short-descriptor translation table format, the Address Space Identifier.

The value of the whole of this register is called the Context ID and is used by:

- The debug logic, for Linked and Unlinked Context ID matching.
- The trace logic, to identify the current process.

The significance of this register is for debug and trace use only.

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch32 System register CONTEXTIDR is architecturally mapped to AArch64 System register [CONTEXTIDR\\_EL1](#).

The register format depends on whether address translation is using the Long-descriptor or the Short-descriptor translation table format.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CONTEXTIDR is a 32-bit register.

## Field descriptions

The CONTEXTIDR bit assignments are:

### When TTBCR.EAE==0:

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

#### PROCID, bits [31:8]

Process Identifier. This field must be programmed with a unique value that identifies the current process.

#### ASID, bits [7:0]

Address Space Identifier. This field is programmed with the value of the current ASID.

### When TTBCR.EAE==1:

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

#### PROCID, bits [31:0]

Process Identifier. This field must be programmed with a unique value that identifies the current process.

## Accessing the CONTEXTIDR

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>, c13, c0, 1	000	001	1101	1111	0000

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	
EL3 using AArch32	x	0	1	-	RW	RW	RW	CONTEXTIDR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	CONTEXTIDR_ns
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	CONTEXTIDR_s
EL3 not implemented	x	x	0	-	RW	n/a	n/a	CONTEXTIDR
EL3 not implemented	x	0	1	-	RW	RW	n/a	CONTEXTIDR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	CONTEXTIDR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	CONTEXTIDR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	CONTEXTIDR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	CONTEXTIDR

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.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.T13==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.T13==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.T13==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.





# CPACR, Architectural Feature Access Control Register

The CPACR characteristics are:

## Purpose

Controls access to trace, and to Advanced SIMD and floating-point functionality from EL0, EL1, and EL3.

In an implementation that includes EL2, the CPACR has no effect on instructions executed at EL2.

This register is part of the Other system 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 CPACR is architecturally mapped to AArch64 System register [CPACR\\_EL1](#).

Bits in the [NSACR](#) control Non-secure access to the CPACR fields. See the field descriptions for more information.

### Note

In the register field descriptions, controls are described as applying at specified Privilege levels. This is because, in Secure state, a PL1 control:

- Applies to execution in a Secure EL3 mode when EL3 is using AArch32.
- Applies to execution in a Secure EL1 mode when EL3 is using AArch64.

See 'Security state, Exception levels, and AArch32 execution privilege' in the ARMv8 ARM, section G1.7.

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. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CPACR is a 32-bit register.

## Field descriptions

The CPACR 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
<a href="#">ASEDIS</a>	0	0	<a href="#">TRCDIS</a>	0	0	0	0	<a href="#">cp11</a>	<a href="#">cp10</a>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### ASEDIS, bit [31]

Disables PL0 and PL1 execution of Advanced SIMD instructions.

ASEDIS	Meaning
0	This control permits execution of Advanced SIMD instructions at PL0 and PL1.
1	All instruction encodings that are Advanced SIMD instruction encodings, but are not also floating-point instruction encodings, are UNDEFINED at PL0 and PL1.

If the implementation does not include Advanced SIMD and floating-point functionality, this field is RES0. Otherwise, it is IMPLEMENTATION DEFINED whether this field is implemented as a RW field. If it is not implemented as a RW field, it is RAZ/WI.

If EL3 is implemented and is using AArch32, and the value of [NSACR.NSASEDIS](#) is 1, this field behaves as RAO/WI in Non-secure state, regardless of its actual value. This applies even if the field is implemented as RAZ/WI.

For the list of instructions affected by this field, see 'Controls of Advanced SIMD operation that do not apply to floating-point operation' in the ARMv8 ARM, section E1.

See the description of CPACR.cp10 for a list of other controls that can disable or trap execution of Advanced SIMD instructions in AArch32 state.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### Bits [30:29]

Reserved, RES0.

#### TRCDIS, bit [28]

Traps PL0 and PL1 System register accesses to all implemented trace registers to Undefined mode.

TRCDIS	Meaning
0	This control has no effect on PL0 and PL1 System register accesses to trace registers.
1	PL0 and PL1 System register accesses to all implemented trace registers are trapped to Undefined mode.

If the implementation does not include a trace macrocell, or does not include a System register interface to the trace macrocell registers, this field is RES0. Otherwise, it is IMPLEMENTATION DEFINED whether this field is implemented as a RW field. If it is not implemented as a RW field, it is RAZ/WI.

If EL3 is implemented and is using AArch32, and the value of [NSACR.NSTRCDIS](#) is 1, this field behaves as RAO/WI in Non-secure state, regardless of its actual value. This applies even if the field is implemented as RAZ/WI.

#### Note

- The ETMv4 architecture does not permit EL0 to access the trace registers. If the implementation includes an ETMv4 implementation, EL0 accesses to the trace registers are UNDEFINED.
- The architecture does not provide traps on trace register accesses through the optional memory-mapped external debug interface.

System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.

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 [27:24]

Reserved, RES0.

#### cp11, bits [23:22]

The value of this field is ignored. If this field is programmed with a different value to the cp10 field then this field is UNKNOWN on a direct read of the CPACR.

If the implementation does not include Advanced SIMD and floating-point functionality, this field is RES0.

In Non-secure state, if EL3 is implemented and is using AArch32, when the value of [NSACR.cp10](#) is 0, this field behaves as RAZ/WI, regardless of its actual value.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### cp10, bits [21:20]

Defines the access rights for the floating-point and Advanced SIMD functionality. Possible values of the field are:

cp10	Meaning
00	PL0 and PL1 accesses to floating-point and Advanced SIMD registers or instructions are UNDEFINED.
01	PL0 accesses to floating-point and Advanced SIMD registers or instructions are UNDEFINED.
10	Reserved. The effect of programming this field to this value is CONSTRAINED UNPREDICTABLE. See 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARMv8 ARM, section J1.1.11.
11	This control permits full access to the floating-point and Advanced SIMD functionality from PL0 and PL1.

The floating-point and Advanced SIMD features controlled by these fields are:

- Execution of any floating-point or Advanced SIMD instruction.
- Any access to the Advanced SIMD and floating-point registers D0-D31 and their views as S0-S31 and Q0-Q15.
- Any access to the [FPSCR](#), [FPSID](#), [MVFR0](#), [MVFR1](#), [MVFR2](#), or [FPEXC](#) System registers.

#### Note

The [CPACR](#) has no effect on floating-point and Advanced SIMD accesses from PL2. These can be disabled by the [HCPTR](#).TCP10 field.

If the implementation does not include Advanced SIMD and floating-point functionality, this field is RES0.

In Non-secure state, if EL3 is implemented and is using AArch32, when the value of [NSACR](#).cp10 is 0, this field behaves as RAZ/WI, regardless of its actual value.

Execution of floating-point and Advanced SIMD instructions in AArch32 state can be disabled or trapped by the following controls:

- CPACR.cp10, or, if executing at EL0, [CPACR\\_EL1](#).FPEN.
- [FPEXC](#).EN.
- If executing in Non-secure state:
  - [HCPTR](#).TCP10, or if EL2 is using AArch64, [CPTR\\_EL2](#).TFP.
  - [NSACR](#).cp10, or if EL3 is using AArch64, [CPTR\\_EL3](#).TFP.
- For Advanced SIMD instructions only:
  - CPACR.ASEDIS.
  - If executing in Non-secure state, [HCPTR](#).TASE and [NSACR](#).NSTRCDIS.

See the descriptions of the controls for more information.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### Bits [19:0]

Reserved, RES0.

## Accessing the CPACR

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, 2	000	010	0001	1111	0000

## 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<sup>®</sup> 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [CPTR\\_EL2](#).TCPAC==1, Non-secure 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- If [CPTR\\_EL2](#).TCPAC==1, Non-secure 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  $\text{SCR\_EL3.NS}=1$  :

- If [HCPTR](#).TCPAC==1, Non-secure 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.

When EL3 is implemented and is using AArch64 :

- If [CPTR\\_EL3](#).TCPAC==1, accesses to this register from EL1 and EL2 are trapped to EL3.

# CPSR, Current Program Status Register

The CPSR characteristics are:

## Purpose

Holds PE status and control information.

This register is part of the Process state registers functional group.

## Usage constraints

The CPSR can be read using the MRS instruction and written using the MSR (immediate) or MSR (register) instructions. For more details on the instruction syntax, see 'PSTATE and banked register access instructions' in the ARMv8 ARM, section F1.5.

## Traps and Enables

There are no traps or enables affecting this register.

## Configuration

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

## Attributes

CPSR is a 32-bit register.

## Field descriptions

The CPSR 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
N	Z	C	V	Q	IT[1:0]	J	0	PAN	0	0		GE			IT[7:2]			E	A	I	F	T	1		M[3:0]						

### N, bit [31]

Negative condition flag. Set to bit[31] of the result of the last flag-setting instruction. If the result is regarded as a two's complement signed integer, then N is set to 1 if the result was negative, and N is set to 0 if the result was positive or zero.

### Z, bit [30]

Zero condition flag. Set to 1 if the result of the last flag-setting instruction was zero, and to 0 otherwise. A result of zero often indicates an equal result from a comparison.

### C, bit [29]

Carry condition flag. Set to 1 if the last flag-setting instruction resulted in a carry condition, for example an unsigned overflow on an addition.

### V, bit [28]

Overflow condition flag. Set to 1 if the last flag-setting instruction resulted in an overflow condition, for example a signed overflow on an addition.

**Q, bit [27]**

Cumulative saturation bit. Set to 1 to indicate that overflow or saturation occurred in some instructions.

**IT[1:0], bits [26:25]**

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]**

In ARMv8.2 and ARMv8.1:

Privileged Access Never. When ARMv8.1-PAN is implemented, defined values are:

PAN	Meaning
0	The translation system is the same as ARMv8.0.
1	Disables privileged read and write accesses to addresses accessible at EL0.

The value of this bit is usually preserved on taking an exception, except in the following situations:

- When the target of the exception is EL1, and the value of the [SCTLR.SPAN](#) bit for the current Security state is 0, this bit is set to 1.
- When the target of the exception is EL3, from Secure state, and the value of the Secure [SCTLR.SPAN](#) is 0, this bit is set to 1.
- When the target of the exception is EL3, from Non-secure state, this bit is set to 0 regardless of the value of the Secure [SCTLR.SPAN](#) bit.

When ARMv8.1-PAN is not implemented, this bit is RES0.

In ARMv8.0:

Reserved, RES0.

**Bits [21:20]**

Reserved, RES0.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

<b>E</b>	<b>Meaning</b>
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTLR.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

**A, bit [8]**

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

<b>A</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**I, bit [7]**

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

<b>I</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**F, bit [6]**

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

<b>F</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**T, bit [5]**

T32 Instruction set state bit. Indicates the AArch32 instruction set state. Possible values of this bit are:

<b>T</b>	<b>Meaning</b>
0	A32 state.
1	T32 state.

**Bit [4]**

Reserved, RES1.

**M[3:0], bits [3:0]**

Current PE mode. Possible values are:

<b>M[3:0]</b>	<b>Mode</b>
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor
0b0111	Abort
0b1010	Hyp
0b1011	Undefined
0b1111	System

Other values are reserved.

02/05/2017 15:43

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



# CSSELR, Cache Size Selection Register

The CSSELR characteristics are:

## Purpose

Selects the current Cache Size ID Register, [CCSIDR](#), 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

AArch32 System register CSSELR is architecturally mapped to AArch64 System register [CSSELR\\_ELI](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CSSELR is a 32-bit register.

## Field descriptions

The CSSELR 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	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.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.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

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, 2, <Rt>, c0, c0, 0	010	000	0000	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	CSSELR
EL3 not implemented	x	0	1	-	RW	RW	n/a	CSSELR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	CSSELR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	CSSELR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	CSSELR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	CSSELR
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	CSSELR_s
EL3 using AArch32	x	0	1	-	RW	RW	RW	CSSELR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	CSSELR_ns

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.TID2](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.
- 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 [HCR\\_EL2.TID2](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.
- 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 [HCR.TID2](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.
- If [HSTR.T0](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.



# CTR, Cache Type Register

The CTR characteristics are:

## Purpose

Provides information about the architecture of the caches.

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 CTR is architecturally mapped to AArch64 System register [CTR\\_EL0](#).

## Attributes

CTR is a 32-bit register.

## Field descriptions

The CTR 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	
1	0	0	0		CWG				ERG				DminLine	L1lp	0	0	0	0	0	0	0	0	0	0	0	0						IminLine

### Bit [31]

Reserved, RES1.

### Bits [30:28]

Reserved, RES0.

### CWG, bits [27:24]

Cache writeback granule. Log<sub>2</sub> of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified.

A value of 0b0000 indicates that this register does not provide Cache writeback granule information and either:

- The architectural maximum of 512 words (2KB) must be assumed.
- The Cache writeback granule can be determined from maximum cache line size encoded in the Cache Size ID Registers.

Values greater than 0b1001 are reserved.

ARM recommends that an implementation that does not support cache write-back implements this field as 0b0001. This applies, for example, to an implementation that supports only write-through caches.

### ERG, bits [23:20]

Exclusives reservation granule. Log<sub>2</sub> of the number of words of the maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions.

A value of 0b0000 indicates that this register does not provide Exclusives reservation granule information and the architectural maximum of 512 words (2KB) must be assumed.

Values greater than 0b1001 are reserved.

### DminLine, bits [19:16]

Log<sub>2</sub> of the number of words in the smallest cache line of all the data caches and unified caches that are controlled by the PE.

### L1Ip, bits [15:14]

Level 1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache. Possible values of this field are:

L1Ip	Meaning
00	VMID aware Physical Index, Physical tag (VPIPT)
01	ASID-tagged Virtual Index, Virtual Tag (AIVIVT)
10	Virtual Index, Physical Tag (VIPT)
11	Physical Index, Physical Tag (PIPT)

The value 0b01 is reserved in ARMv8.

The value 0b00 is permitted only in an implementation that includes ARMv8.2-PIPTV, otherwise the value is reserved.

### Bits [13:4]

Reserved, RES0.

### IminLine, bits [3:0]

Log<sub>2</sub> of the number of words in the smallest cache line of all the instruction caches that are controlled by the PE.

## Accessing the CTR

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, 1	000	001	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<sup>®</sup> 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](#).TID2==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](#).TID2==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](#).TID2==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.

02/05/2017 15:43

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

# DACR, Domain Access Control Register

The DACR characteristics are:

## Purpose

Defines the access permission for each of the sixteen memory domains.

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch32 System register DACR is architecturally mapped to AArch64 System register [DACR32\\_EL2](#).

When EL3 is using AArch32, write access to DACR(S) is disabled when the CP15SDISABLE signal is asserted HIGH.

This register has no function when [TTBCR](#).EAE is set to 1, to select the Long-descriptor translation table format.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

DACR is a 32-bit register.

## Field descriptions

The DACR 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
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0																

**D<n>, bits [2n+1:2n], for n = 0 to 15**

Domain n access permission, where n = 0 to 15. Permitted values are:

D<n>	Meaning
00	No access. Any access to the domain generates a Domain fault.
01	Client. Accesses are checked against the permission bits in the translation tables.
11	Manager. Accesses are not checked against the permission bits in the translation tables.

The value 10 is reserved.

## Accessing the DACR

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>, c3, c0, 0	000	000	0011	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	DACR
EL3 not implemented	x	0	1	-	RW	RW	n/a	DACR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	DACR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	DACR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	DACR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	DACR
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	DACR_s
EL3 using AArch32	x	0	1	-	RW	RW	RW	DACR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	DACR_ns

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

When EL3 is using AArch32, write access to DACR\_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.T3==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.T3==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.T3==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.



# DBGAUTHSTATUS, Debug Authentication Status register

The DBGAUTHSTATUS characteristics are:

## Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

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 DBGAUTHSTATUS is architecturally mapped to AArch64 System register [DBGAUTHSTATUS\\_EL1](#).

AArch32 System register DBGAUTHSTATUS is architecturally mapped to External register [DBGAUTHSTATUS\\_EL1](#).

This register is required in all implementations.

## Attributes

DBGAUTHSTATUS is a 32-bit register.

## Field descriptions

The DBGAUTHSTATUS 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	SNID	SID	NSNID	NSID				

### Bits [31:8]

Reserved, RES0.

### SNID, bits [7:6]

Secure non-invasive debug. Possible values of this field are:

SNID	Meaning
00	Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state.
10	Implemented and disabled. ExternalSecureNoninvasiveDebugEnabled() == FALSE.
11	Implemented and enabled. ExternalSecureNoninvasiveDebugEnabled() == TRUE.

Other values are reserved.

### SID, bits [5:4]

Secure invasive debug. Possible values of this field are:

SID	Meaning
00	Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state.
10	Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE.
11	Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.

Other values are reserved.

**NSNID, bits [3:2]**

Non-secure non-invasive debug. Possible values of this field are:

NSNID	Meaning
00	Not implemented. EL3 is not implemented and the implemented Security state is Secure state.
10	Implemented and disabled. ExternalNoninvasiveDebugEnabled() == FALSE.
11	Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE.

Other values are reserved.

**NSID, bits [1:0]**

Non-secure invasive debug. Possible values of this field are:

NSID	Meaning
00	Not implemented. EL3 is not implemented and the implemented Security state is Secure state.
10	Implemented and disabled. ExternalInvasiveDebugEnabled() == FALSE.
11	Implemented and enabled. ExternalInvasiveDebugEnabled() == TRUE.

Other values are reserved.

**Accessing the DBGAUTHSTATUS**

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
p14, 0, <Rt>, c7, c14, 6	000	110	0111	1110	1110

**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<sup>®</sup> 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 :

- If [MDCR\\_EL2](#).TDA==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 [HDCR](#).TDA==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDA==1, read accesses to this register from EL1 and EL2 are trapped to EL3.

02/05/2017 15:43

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

# 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 <a href="#">CONTEXTIDR_EL1</a> match (introduced in ARMv8.1).
0111	Linked <a href="#">CONTEXTIDR_EL1</a> 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 <a href="#">CONTEXTIDR_EL2</a> match (introduced in ARMv8.1).
1101	Linked <a href="#">CONTEXTIDR_EL2</a> 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	<a href="#">DBGBVR&lt;n&gt;</a>	Use for T32 instructions.
1100	<a href="#">DBGBVR&lt;n&gt;+2</a>	Use for T32 instructions.
1111	<a href="#">DBGBVR&lt;n&gt;</a>	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	<a href="#">DBGBVR&lt;n&gt;</a>	Use for stepping T32 instructions.
1100	<a href="#">DBGBVR&lt;n&gt;+2</a>	Use for stepping T32 instructions.
1111	<a href="#">DBGBVR&lt;n&gt;</a>	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, accesses to this register from PL1 and PL2 are trapped to Debug 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.

02/05/2017 15:43

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



# 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 [DBGXVR<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]

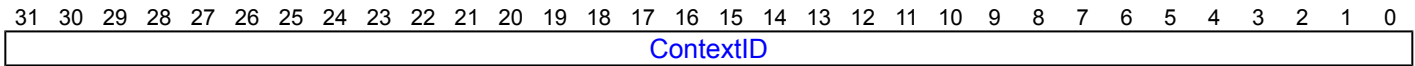
Bits[31:2] of the address value for comparison.

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.

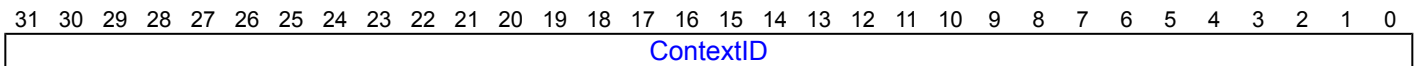
The value is compared against [CONTEXTIDR](#) in the following cases:

- 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:

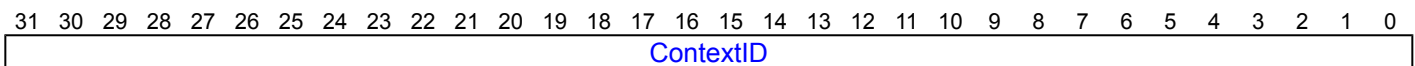


### ContextID, bits [31:0]

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, accesses to this register from PL1 and PL2 are trapped to Debug 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.

02/05/2017 15:43

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

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

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

AArch32 System register DBGBXVR<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 DBGBXVR<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 ARMv8.2 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 DBGBCR<n>.BT==0b11xx 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
ContextID2																															

#### 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 DBGBXVR<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, accesses to this register from PL1 and PL2 are trapped to Debug 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.

02/05/2017 15:43

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

# DBGCLAIMCLR, Debug Claim Tag Clear register

The DBGCLAIMCLR characteristics are:

## Purpose

Used by software to read the values of the CLAIM tag bits, and to clear these bits to 0.

The architecture does not define any functionality for the CLAIM tag bits.

---

### Note

CLAIM tags are typically used for communication between the debugger and target software.

---

Used in conjunction with the [DBGCLAIMSET](#) register.

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 DBGCLAIMCLR is architecturally mapped to AArch64 System register [DBGCLAIMCLR\\_EL1](#).

AArch32 System register DBGCLAIMCLR is architecturally mapped to External register [DBGCLAIMCLR\\_EL1](#).

An implementation must include 8 CLAIM tag bits.

This register is in the Cold reset domain. See the CLAIM field description for the effect of a Cold reset on the value returned by this register. This register is not affected by a Warm reset.

## Attributes

DBGCLAIMCLR is a 32-bit register.

## Field descriptions

The DBGCLAIMCLR 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	CLAIM									

[CLAIM](#)

### Bits [31:8]

Reserved, RAZ/SBZ. Software can rely on these bits reading as zero, and must use a should-be-zero policy on writes. Implementations must ignore writes.

### CLAIM, bits [7:0]

Read or clear CLAIM tag bits. Reading this field returns the current value of the CLAIM tag bits.

Writing a 1 to one of these bits clears the corresponding CLAIM tag bit to 0. This is an indirect write to the CLAIM tag bits. A single write operation can clear multiple CLAIM tag bits to 0.

Writing 0 to one of these bits has no effect.

A cold reset clears the CLAIM tag bits to 0.

## Accessing the DBGCLAIMCLR

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>, c7, c9, 6	000	110	0111	1110	1001

## 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<sup>®</sup> 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 :

- 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.



# DBGCLAIMSET, Debug Claim Tag Set register

The DBGCLAIMSET characteristics are:

## Purpose

Used by software to set the CLAIM tag bits to 1.

The architecture does not define any functionality for the CLAIM tag bits.

---

### Note

CLAIM tags are typically used for communication between the debugger and target software.

---

Used in conjunction with the [DBGCLAIMCLR](#) register.

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 DBGCLAIMSET is architecturally mapped to AArch64 System register [DBGCLAIMSET\\_EL1](#).

AArch32 System register DBGCLAIMSET is architecturally mapped to External register [DBGCLAIMSET\\_EL1](#).

An implementation must include 8 CLAIM tag bits.

## Attributes

DBGCLAIMSET is a 32-bit register.

## Field descriptions

The DBGCLAIMSET 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								

CLAIM

### Bits [31:8]

Reserved, RAZ/SBZ. Software can rely on these bits reading as zero, and must use a should-be-zero policy on writes. Implementations must ignore writes.

### CLAIM, bits [7:0]

Set CLAIM tag bits. RAO.

Writing a 1 to one of these bits sets the corresponding CLAIM tag bit to 1. This is an indirect write to the CLAIM tag bits. A single write operation can set multiple CLAIM tag bits to 1.

Writing 0 to one of these bits has no effect.

A cold reset clears the CLAIM tag bits to 0.

## Accessing the DBGCLAIMSET

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>, c7, c8, 6	000	110	0111	1110	1000

## 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<sup>®</sup> 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 :

- 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.

# DBGDCCINT, DCC Interrupt Enable Register

The DBGDCCINT characteristics are:

## Purpose

Enables interrupt requests to be signaled based on the DCC status flags.

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 DBGDCCINT is architecturally mapped to AArch64 System register [MDCCINT\\_EL1](#).

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

DBGDCCINT is a 32-bit register.

## Field descriptions

The DBGDCCINT 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	RX	TX	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

### Bit [31]

Reserved, RES0.

### RX, bit [30]

DCC interrupt request enable control for DTRRX. Enables a common COMMIRQ interrupt request to be signaled based on the DCC status flags.

RX	Meaning
0	No interrupt request generated by DTRRX.
1	Interrupt request will be generated on RXfull == 1.

If legacy COMMRX and COMMTX signals are implemented, then these are not affected by the value of this bit.

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

### TX, bit [29]

DCC interrupt request enable control for DTRTX. Enables a common COMMIRQ interrupt request to be signaled based on the DCC status flags.

TX	Meaning
0	No interrupt request generated by DTRTX.
1	Interrupt request will be generated on TXfull == 0.

If legacy COMMRX and COMMTX signals are implemented, then these are not affected by the value of this bit.

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

## Bits [28:0]

Reserved, RES0.

## Accessing the DBGDCCINT

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, c2, 0	000	000	0000	1110	0010

## 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<sup>®</sup> 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 :

- 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.

# DBGDEVID, Debug Device ID register 0

The DBGDEVID characteristics are:

## Purpose

Adds to the information given by the [DBGDIDR](#) by describing other features of the debug implementation.

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.

This register is required in all implementations.

## Attributes

DBGDEVID is a 32-bit register.

## Field descriptions

The DBGDEVID 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
<a href="#">CIDMask</a>				<a href="#">AuxRegs</a>				<a href="#">DoubleLock</a>				<a href="#">VirtExtns</a>				<a href="#">VectorCatch</a>				<a href="#">BPAAddrMask</a>			<a href="#">WPAAddrMask</a>					<a href="#">PCSample</a>			

### CIDMask, bits [31:28]

Indicates the level of support for the Context ID matching breakpoint masking capability. Permitted values of this field are:

CIDMask	Meaning
0000	Context ID masking is not implemented.
0001	Context ID masking is implemented.

All other values are reserved. The value of this for ARMv8 is 0000.

### AuxRegs, bits [27:24]

Indicates support for Auxiliary registers. Permitted values for this field are:

AuxRegs	Meaning
0000	None supported.
0001	Support for External Debug Auxiliary Control Register, <a href="#">EDACR</a> .

All other values are reserved.

### DoubleLock, bits [23:20]

Indicates the presence of the [DBGOSDLR](#), OS Double Lock Register. Permitted values of this field are:

DoubleLock	Meaning
0000	The <a href="#">DBGOSDLR</a> is not present.
0001	The <a href="#">DBGOSDLR</a> is present.

All other values are reserved. The value of this for ARMv8 is 0001.

**VirtExtns, bits [19:16]**

Indicates whether EL2 is implemented. Permitted values of this field are:

<b>VirtExtns</b>	<b>Meaning</b>
0000	EL2 is not implemented.
0001	EL2 is implemented.

All other values are reserved.

**VectorCatch, bits [15:12]**

Defines the form of Vector Catch exception implemented. Permitted values of this field are:

<b>VectorCatch</b>	<b>Meaning</b>
0000	Address matching Vector Catch exception implemented.
0001	Exception matching Vector Catch exception implemented.

All other values are reserved.

**BPAAddrMask, bits [11:8]**

Indicates the level of support for the instruction address matching breakpoint masking capability. Permitted values of this field are:

<b>BPAAddrMask</b>	<b>Meaning</b>
0000	Breakpoint address masking might be implemented. If not implemented, <a href="#">DBGBCR&lt;n&gt;[28:24]</a> is RAZ/WI.
0001	Breakpoint address masking is implemented.
1111	Breakpoint address masking is not implemented. <a href="#">DBGBCR&lt;n&gt;[28:24]</a> is RES0.

All other values are reserved. The value of this for ARMv8 is 1111.

**WPAAddrMask, bits [7:4]**

Indicates the level of support for the data address matching watchpoint masking capability. Permitted values of this field are:

<b>WPAAddrMask</b>	<b>Meaning</b>
0000	Watchpoint address masking might be implemented. If not implemented, <a href="#">DBGWCR&lt;n&gt;.MASK</a> (Address mask) is RAZ/WI.
0001	Watchpoint address masking is implemented.
1111	Watchpoint address masking is not implemented. <a href="#">DBGWCR&lt;n&gt;.MASK</a> (Address mask) is RES0.

All other values are reserved. The value of this for ARMv8 is 0001.

**PCSample, bits [3:0]**

Indicates the level of PC Sample-based Profiling support using external debug registers. Permitted values of this field are:

<b>PCSample</b>	<b>Meaning</b>
0000	Architecture-defined form of PC Sample-based Profiling not implemented using external debug registers.
0010	Only <a href="#">EDPCSR</a> and <a href="#">EDCIDS</a> are implemented. This option is only permitted if EL3 and EL2 are not implemented.
0011	<a href="#">EDPCSR</a> , <a href="#">EDCIDS</a> , and <a href="#">EDVIDSR</a> are implemented.

All other values are reserved.

From ARMv8.2 onwards, the only permitted value is 0b0000. The architecture defines the functionality in a different set of registers, see [PMDEVID](#).

## Accessing the DBGDEVID

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
p14, 0, <Rt>, c7, c2, 7	000	111	0111	1110	0010

## 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<sup>®</sup> 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 :

- If [MDCR\\_EL2](#).TDA==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 [HDCR](#).TDA==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDA==1, read accesses to this register from EL1 and EL2 are trapped to EL3.

02/05/2017 15:43

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

# DBGDEVID1, Debug Device ID register 1

The DBGDEVID1 characteristics are:

## Purpose

Adds to the information given by the [DBGDIDR](#) by describing other features of the debug implementation.

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.

This register is required in all implementations.

## Attributes

DBGDEVID1 is a 32-bit register.

## Field descriptions

The DBGDEVID1 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	0

[PCSROffset](#)

### Bits [31:4]

Reserved, RES0.

### PCSROffset, bits [3:0]

This field indicates the offset applied to PC samples returned by reads of [EDPCSR](#). Permitted values of this field in ARMv8 are:

PCSROffset	Meaning
0000	<a href="#">EDPCSR</a> not implemented.
0010	<a href="#">EDPCSR</a> implemented. Samples have no offset applied and do not sample the instruction set state in AArch32 state.

**Note**

In ARMv7, a PCSROffset value of 0000 has an alternative meaning that [EDPCSR](#) is implemented and returns values that have an offset applied and indicate the Instruction set state. This implementation option is not permitted in ARMv8.

From ARMv8.2 onwards, the only permitted value is 0b0000. The architecture defines the functionality in a different set of registers, see [PMDEVID](#).

## Accessing the DBGDEVID1

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
p14, 0, <Rt>, c7, c1, 7	000	111	0111	1110	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<sup>®</sup> 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 :

- If [MDCR\\_EL2](#).TDA==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 [HDCR](#).TDA==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDA==1, read accesses to this register from EL1 and EL2 are trapped to EL3.

# DBGDEVID2, Debug Device ID register 2

The DBGDEVID2 characteristics are:

## Purpose

Reserved for future descriptions of features of the debug implementation.

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.

## Attributes

DBGDEVID2 is a 32-bit register.

## Field descriptions

The DBGDEVID2 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	0

### Bits [31:0]

Reserved, RES0.

## Accessing the DBGDEVID2

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
p14, 0, <Rt>, c7, c0, 7	000	111	0111	1110	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 :

- If [MDCR\\_EL2](#).TDA==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 [HDCR](#).TDA==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDA==1, read accesses to this register from EL1 and EL2 are trapped to EL3.

02/05/2017 15:43

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

# DBGDIDR, Debug ID Register

The DBGDIDR characteristics are:

## Purpose

Specifies which version of the Debug architecture is implemented, and some features of the debug implementation.

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.

If EL1 cannot use AArch32 then the implementation of this register is OPTIONAL and deprecated.

## Attributes

DBGDIDR is a 32-bit register.

## Field descriptions

The DBGDIDR 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
WRPs				BRPs				CTX_CMPs				Version	1	nSUHD_imp	0	SE_imp	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### WRPs, bits [31:28]

The number of watchpoints implemented, minus 1.

Permitted values of this field are from 0b0001 for 2 implemented watchpoints, to 0b1111 for 16 implemented watchpoints.

The value of 0b0000 is reserved.

If AArch64 is implemented, this field has the same value as [ID\\_AA64DFR0\\_EL1](#).WRPs.

### BRPs, bits [27:24]

The number of breakpoints implemented, minus 1.

Permitted values of this field are from 0b0001 for 2 implemented breakpoint, to 0b1111 for 16 implemented breakpoints.

The value of 0b0000 is reserved.

If AArch64 is implemented, this field has the same value as [ID\\_AA64DFR0\\_EL1](#).BRPs.

### CTX\_CMPs, bits [23:20]

The number of breakpoints that can be used for Context matching, minus 1.

Permitted values of this field are from 0b0000 for 1 Context matching breakpoint, to 0b1111 for 16 Context matching breakpoints.

The Context matching breakpoints must be the highest addressed breakpoints. For example, if six breakpoints are implemented and two are Context matching breakpoints, they must be breakpoints 4 and 5.

If AArch64 is implemented, this field has the same value as [ID\\_AA64DFR0\\_EL1](#).CTX\_CMPs.

**Version, bits [19:16]**

The Debug architecture version. Defined values are:

Version	Meaning
0001	ARMv6, v6 Debug architecture.
0010	ARMv6, v6.1 Debug architecture.
0011	ARMv7, v7 Debug architecture, with baseline CP14 registers implemented.
0100	ARMv7, v7 Debug architecture, with all CP14 registers implemented.
0101	ARMv7, v7.1 Debug architecture.
0110	ARMv8, v8 Debug architecture.
0111	ARMv8.1, v8 Debug architecture, with Virtualization Host Extensions.
1000	ARMv8.2, v8.2 Debug architecture.

All other values are reserved.

- In an ARMv8.0 implementation, the only permitted value is 0110.
- In an ARMv8.1 implementation that includes ARMv8.1-VHE, the only permitted value is 0111.
- In an ARMv8.1 implementation that does not include ARMv8.1-VHE, the permitted values are 0110 and 0111.
- In an ARMv8.2 implementation, the only permitted value is 1000.

**Bit [15]**

Reserved, RES1.

**nSUHD\_imp, bit [14]**

In ARMv7-A, was Secure User Halting Debug not implemented.

The value of this bit must match the value of the SE\_imp bit.

**Bit [13]**

Reserved, RES0.

**SE\_imp, bit [12]**

EL3 implemented. The meanings of the values of this bit are:

SE_imp	Meaning
0	EL3 not implemented.
1	EL3 implemented.

The value of this bit must match the value of the nSUHD\_imp bit.

**Bits [11:0]**

Reserved, RES0.

**Accessing the DBGDIDR**

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
p14, 0, <Rt>, c0, c0, 0	000	000	0000	1110	0000

## 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.

ARM deprecates any access to this register from EL0.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 [DBGDSCRExt](#).UDCCdis==1, read accesses to this register from EL0 are trapped to Undefined mode.
- If [MDSCR\\_EL1](#).TDCC==1, 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](#).TDA==1, Non-secure read 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](#).TDA==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDA==1, read accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

# DBGDRAR, Debug ROM Address Register

The DBGDRAR characteristics are:

## Purpose

Defines the base physical address of a 4KB-aligned memory-mapped debug component, usually a ROM table that locates and describes the memory-mapped debug components in the system. ARMv8 deprecates any use of this register.

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 DBGDRAR is architecturally mapped to AArch64 System register [MDRAR\\_EL1](#).

If EL1 cannot use AArch32 then the implementation of this register is OPTIONAL and deprecated.

## Attributes

DBGDRAR is a 64-bit register that can also be accessed as a 32-bit value. If it is accessed as a 32-bit register, bits [31:0] are read.

## Field descriptions

The DBGDRAR 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					
ROMADDR[31:12]																				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Valid

### ROMADDR[31:12], bits [31:12]

Bits[31:12] of the ROM table physical address. Bits [11:0] of the address are zero.

In an implementation that includes EL3, ROMADDR is an address in Non-secure memory. It is IMPLEMENTATION DEFINED whether the ROM table is also accessible in Secure memory.

### Bits [11:2]

Reserved, RES0.

### Valid, bits [1:0]

This field indicates whether the ROM Table address is valid. The permitted values of this field are:

Valid	Meaning
00	ROM Table address is not valid. Software must ignore ROMADDR.
11	ROM Table address is valid.

Other values are reserved.

**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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ROMADDR[47:12]															
ROMADDR[47:12]																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Valid
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**Bits [63:48]**

Reserved, RES0.

**ROMADDR[47:12], bits [47:12]**

Bits[47:12] of the ROM table physical address.

If the physical address size in bits (PAsize) is less than 48 then the register bits corresponding to ROMADDR [47:PAsize] are RES0.

Bits [11:0] of the ROM table physical address are zero.

ARM strongly recommends that bits ROMADDR[(PAsize-1):32] are zero in any system that supports AArch32 at the highest implemented Exception level.

In an implementation that includes EL3, ROMADDR is an address in Non-secure memory. It is IMPLEMENTATION DEFINED whether the ROM table is also accessible in Secure memory.

**Bits [11:2]**

Reserved, RES0.

**Valid, bits [1:0]**

This field indicates whether the ROM Table address is valid. The permitted values of this field are:

Valid	Meaning
00	ROM Table address is not valid. Software must ignore ROMADDR.
11	ROM Table address is valid.

Other values are reserved.

**Accessing the DBGDRAR**

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
p14, 0, <Rt>, c1, c0, 0	000	000	0001	1110	0000

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

MRRC <syntax>

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

<syntax>	opc1	coproc	CRm
p14, 0, <Rt>, <Rt2>, c1	0000	1110	0001

**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<sup>®</sup> 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 [DBGDSCRExt](#).UDCCdis==1, read accesses to this register from EL0 are trapped to Undefined mode.
- If [MDSCR\\_EL1](#).TDCC==1, 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](#).TDRA==1, Non-secure read 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](#).TDRA==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# DBGDSAR, Debug Self Address Register

The DBGDSAR characteristics are:

## Purpose

In earlier versions of the ARM Architecture, this register defines the offset from the base address defined in [DBGDRAR](#) of the physical base address of the debug registers for the PE. ARMv8 deprecates any use of this register.

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.

If EL1 cannot use AArch32 then the implementation of this register is OPTIONAL and deprecated.

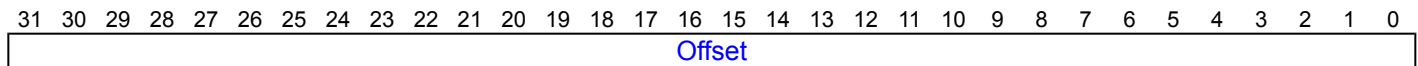
## Attributes

DBGDSAR is a 64-bit register that can also be accessed as a 32-bit value. If it is accessed as a 32-bit register, bits [31:0] are read.

## Field descriptions

The DBGDSAR bit assignments are:

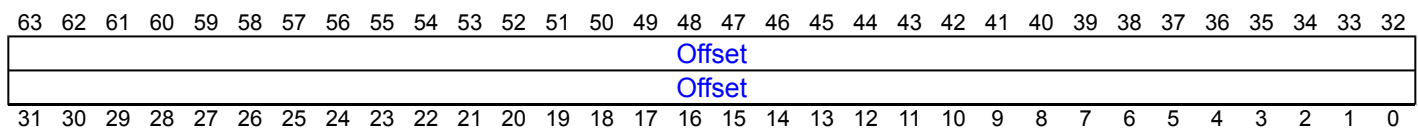
### When accessing as a 32-bit register:



#### Offset, bits [31:0]

This register value is RAZ.

### When accessing as a 64-bit register:



#### Offset, bits [63:0]

This register value is RAZ.

## Accessing the DBGDSAR

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
p14, 0, <Rt>, c2, c0, 0	000	000	0010	1110	0000

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

MRRC <syntax>

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

<syntax>	opc1	coproc	CRm
p14, 0, <Rt>, <Rt2>, c2	0000	1110	0010

## 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<sup>®</sup> 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 [DBGDSCRExt](#).UDCCdis==1, read accesses to this register from EL0 are trapped to Undefined mode.
- If [MDSCR\\_EL1](#).TDCC==1, 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](#).TDRA==1, Non-secure read 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](#).TDRA==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDA==1, read accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

# DBGDSCRext, Debug Status and Control Register, External View

The DBGDSCRext characteristics are:

## Purpose

Main control register for the debug implementation.

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 DBGDSCRext is architecturally mapped to AArch64 System register [MDSCR\\_EL1](#).

This register is required in all implementations.

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

DBGDSCRext is a 32-bit register.

## Field descriptions

The DBGDSCRext 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	RXfull	TXfull	0	RXO	TXU	0	0	INTdis	TDA	0	SC2NS	SPNIDdis	SPIDdis	MDBGGen	HDE	0	UDCCdis	0	0	0	0	0	0	ERR	MOE	0	0				

### Bit [31]

Reserved, RES0.

### RXfull, bit [30]

DTRRX full. Used for save/restore of [EDSCR](#).RXfull.

When [DBGOSLSR](#).OSLK == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [DBGOSLSR](#).OSLK == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR](#).RXfull.

ARM deprecates use of this bit other than for save/restore. Use [DBGDSCRint](#) to access the DTRRX full status.

Reads and writes of this bit are indirect accesses to [EDSCR](#).RXfull.

The architected behavior of this field determines the value it returns after a reset.

### TXfull, bit [29]

DTRTX full. Used for save/restore of [EDSCR](#).TXfull.

When [DBGOSLSR](#).OSLK == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [DBGOSLSR](#).OSLK == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR](#).TXfull.

ARM deprecates use of this bit other than for save/restore. Use [DBGDSCRint](#) to access the DTRTX full status.

Reads and writes of this bit are indirect accesses to [EDSCR.TXfull](#).

The architected behavior of this field determines the value it returns after a reset.

#### Bit [28]

Reserved, RES0.

#### RXO, bit [27]

Used for save/restore of [EDSCR.RXO](#).

When [DBGOSLSR.OSLK](#) == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [DBGOSLSR.OSLK](#) == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR.RXO](#).

Reads and writes of this bit are indirect accesses to [EDSCR.RXO](#).

The architected behavior of this field determines the value it returns after a reset.

#### TXU, bit [26]

Used for save/restore of [EDSCR.TXU](#).

When [DBGOSLSR.OSLK](#) == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [DBGOSLSR.OSLK](#) == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR.TXU](#).

Reads and writes of this bit are indirect accesses to [EDSCR.TXU](#).

The architected behavior of this field determines the value it returns after a reset.

#### Bits [25:24]

Reserved, RES0.

#### INTdis, bits [23:22]

Used for save/restore of [EDSCR.INTdis](#).

When [DBGOSLSR.OSLK](#) == 0 (the OS lock is unlocked), this field is RO, and software must treat it as UNK/SBZP.

When [DBGOSLSR.OSLK](#) == 1 (the OS lock is locked), this field is RW and holds the value of [EDSCR.INTdis](#).

Reads and writes of this field are indirect accesses to [EDSCR.INTdis](#).

The architected behavior of this field determines the value it returns after a reset.

#### TDA, bit [21]

Used for save/restore of [EDSCR.TDA](#).

When [DBGOSLSR.OSLK](#) == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [DBGOSLSR.OSLK](#) == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR.TDA](#).

Reads and writes of this bit are indirect accesses to [EDSCR.TDA](#).

The architected behavior of this field determines the value it returns after a reset.

#### Bit [20]

Reserved, RES0.

**Bit [19]**

In ARMv8.2 and ARMv8.0:

Reserved, RES0.

In ARMv8.1:

Used for save/restore of [EDSCR](#).SC2.When [DBGOSLSR](#).OSLK == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.When [DBGOSLSR](#).OSLK == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR](#).SC2.Reads and writes of this bit are indirect accesses to [EDSCR](#).SC2.

If the PC Sample-based Profiling Extension is not implemented, then this field is RES0.

**NS, bit [18]**

Non-secure status. Returns the inverse of IsSecure(). This bit is RO.

ARM deprecates use of this field.

**SPNIDdis, bit [17]**

Secure privileged profiling disabled status bit. This bit is RO. Permitted values are:

SPNIDdis	Meaning
0	If EL3 is implemented, profiling allowed in Secure privileged modes.
1	If EL3 is implemented, profiling prohibited in Secure privileged modes.

This field is RES0 if EL3 is not implemented.

- This field is RES1 if either:
  - EL3 is using AArch64 and the Effective value of [SCR\\_EL3](#).NS is 1.
  - EL3 is using AArch32 and the Effective value of [SCR](#).NS is 1.
- Otherwise, the field is RES0 if any of the following applies, and RES1 otherwise:
  - ARMv8.2-Debug is not implemented and ExternalSecureNoninvasiveDebugEnabled() returns TRUE.
  - EL3 is using AArch32 and the value of [SDCR](#).SPME is 1.
  - EL3 is using AArch64 and the value of [MDCR\\_EL3](#).SPME is 1.

ARM deprecates use of this field.

**SPIDdis, bit [16]**Secure privileged AArch32 invasive self-hosted debug disabled status bit. This bit is RO and depends on the value of [SDCR](#).SPD and the pseudocode function AArch32.SelfHostedSecurePrivilegedInvasiveDebugEnabled(). Permitted values are:

SPIDdis	Meaning
0	Self-hosted debug enabled in Secure privileged AArch32 modes.
1	Self-hosted debug disabled in Secure privileged AArch32 modes.

This bit reads as 1 if any of the following is true and reads as 0 otherwise:

- [SDCR](#).SPD has the value 10.
- [SDCR](#).SPD has the value 00 and AArch32.SelfHostedSecurePrivilegedInvasiveDebugEnabled() returns FALSE.

ARM deprecates use of this field.

**MDBGGen, bit [15]**

Monitor debug events enable. Enable Breakpoint, Watchpoint, and Vector Catch exceptions.

MDBGEn	Meaning
0	Breakpoint, Watchpoint, and Vector Catch exceptions disabled.
1	Breakpoint, Watchpoint, and Vector Catch exceptions enabled.

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

#### HDE, bit [14]

Used for save/restore of [EDSCR.HDE](#).

When [DBGOSLSR.OSLK](#) == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [DBGOSLSR.OSLK](#) == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR.HDE](#).

Reads and writes of this bit are indirect accesses to [EDSCR.HDE](#).

The architected behavior of this field determines the value it returns after a reset.

#### Bit [13]

Reserved, RES0.

#### UDCCdis, bit [12]

Traps EL0 accesses to the DCC registers to Undefined mode.

UDCCdis	Meaning
0	This control does not cause any instructions to be trapped.
1	EL0 accesses to the <a href="#">DBGDSCRint</a> , <a href="#">DBGDTRRXint</a> , <a href="#">DBGDTRTXint</a> , <a href="#">DBGDIDR</a> , <a href="#">DBGDSAR</a> , and <a href="#">DBGDRAR</a> are trapped to Undefined mode.

#### Note

All accesses to these registers are trapped, including LDC and STC accesses to [DBGDTRTXint](#) and [DBGDTRRXint](#), and MRRC accesses to [DBGDSAR](#) and [DBGDRAR](#).

Traps of EL0 accesses to the [DBGDTRRXint](#) and [DBGDTRTXint](#) are ignored in Debug state.

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

#### Bits [11:7]

Reserved, RES0.

#### ERR, bit [6]

Used for save/restore of [EDSCR.ERR](#).

When [DBGOSLSR.OSLK](#) == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [DBGOSLSR.OSLK](#) == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR.ERR](#).

Reads and writes of this bit are indirect accesses to [EDSCR.ERR](#).

The architected behavior of this field determines the value it returns after a reset.

#### MOE, bits [5:2]

Method of Entry for debug exception. When a debug exception is taken to an Exception level using AArch32, this field is set to indicate the event that caused the exception:

MOE	Meaning
0001	Breakpoint
0011	Software breakpoint (BKPT) instruction
0101	Vector catch
1010	Watchpoint

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 DBGDSCRext

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, c2, 2	000	010	0000	1110	0010

## 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.

Individual fields within this register might have restricted accessibility when [DBGOSLSR.OSLK](#) == 0 (the OS lock is unlocked.) See the field descriptions for more detail.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 :

- 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.





# DBGDSCRint, Debug Status and Control Register, Internal View

The DBGDSCRint characteristics are:

## Purpose

Main control register for the debug implementation. This is an internal, read-only view.

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.

This register is required in all implementations.

DBGDSCRint.{NS, SPNIDdis, SPIDdis, MDBGen, UDCCdis, MOE} are UNKNOWN when the register is accessed at EL0. However, although these values are not accessible at EL0 by instructions that are neither UNPREDICTABLE nor return UNKNOWN values, it is permissible for an implementation to return the values of DBGDSCRext.{NS, SPNIDdis, SPIDdis, MDBGen, UDCCdis, MOE} for these fields at EL0.

It is also permissible for an implementation to return the same values as defined for a read of DBGDSCRint at EL1 or above. (This is the case even if the implementation does not support AArch32 at EL1 or above.)

## Attributes

DBGDSCRint is a 32-bit register.

## Field descriptions

The DBGDSCRint 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	RXfull	TXfull	0	0	0	0	0	0	0	0	0	0	0	NS	SPNIDdis	SPIDdis	MDBGen	0	0	UDCCdis	0	0	0	0	0	0	0	MOE	0	0	

### Bit [31]

Reserved, RES0.

### RXfull, bit [30]

DTRRX full. Read-only view of the equivalent bit in the [EDSCR](#).

### TXfull, bit [29]

DTRTX full. Read-only view of the equivalent bit in the [EDSCR](#).

### Bits [28:19]

Reserved, RES0.

### NS, bit [18]

Non-secure status.

Read-only view of the equivalent bit in the [DBGDSCRext](#). ARM deprecates use of this field.

**SPNIDdis, bit [17]**

Secure privileged non-invasive debug disable.

Read-only view of the equivalent bit in the [DBGDSCRext](#). ARM deprecates use of this field.

**SPIDdis, bit [16]**

Secure privileged invasive debug disable.

Read-only view of the equivalent bit in the [DBGDSCRext](#). ARM deprecates use of this field.

**MDBGen, bit [15]**

Monitor debug events enable.

Read-only view of the equivalent bit in the [DBGDSCRext](#).

**Bits [14:13]**

Reserved, RES0.

**UDCCdis, bit [12]**

User mode access to Debug Communications Channel disable.

Read-only view of the equivalent bit in the [DBGDSCRext](#). ARM deprecates use of this field.

**Bits [11:6]**

Reserved, RES0.

**MOE, bits [5:2]**

Method of Entry for debug exception. When a debug exception is taken to an Exception level using AArch32, this field is set to indicate the event that caused the exception:

MOE	Meaning
0001	Breakpoint
0011	Software breakpoint (BKPT) instruction
0101	Vector catch
1010	Watchpoint

Read-only view of the equivalent bit in the [DBGDSCRext](#).

**Bits [1:0]**

Reserved, RES0.

**Accessing the DBGDSCRint**

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
p14, 0, <Rt>, c0, c1, 0	000	000	0000	1110	0001

## 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 [DBGDSCRext](#).UDCCdis==1, read accesses to this register from EL0 are trapped to Undefined mode.
- If [MDSCR\\_EL1](#).TDCC==1, 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](#).TDA==1, Non-secure read 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](#).TDA==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDA==1, read accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

02/05/2017 15:43

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

# DBGDTRRText, Debug OS Lock Data Transfer Register, Receive, External View

The DBGDTRRText characteristics are:

## Purpose

Used for save/restore of [DBGDTRRXint](#). It is a component of the Debug Communications Channel.

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 DBGDTRRText is architecturally mapped to AArch64 System register [OSDTRRX\\_EL1](#).

## Attributes

DBGDTRRText is a 32-bit register.

## Field descriptions

The DBGDTRRText 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
Update DTRRX without side-effect																															

### Bits [31:0]

Update DTRRX without side-effect.

Writes to this register update the value in DTRRX and do not change RXfull.

Reads of this register return the last value written to DTRRX and do not change RXfull.

For the full behavior of the Debug Communications Channel, see 'The Debug Communication Channel and Instruction Transfer Register' in the ARM ARM, chapter H4.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN on Cold reset. This field is not affected on Warm reset.

## Accessing the DBGDTRRText

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, c0, 2	000	010	0000	1110	0000

## 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.

ARM deprecates reads and writes of DBGDTRRText through the System register interface when the OS lock is unlocked.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 :

- 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.

02/05/2017 15:43

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

# DBGDTRRXint, Debug Data Transfer Register, Receive

The DBGDTRRXint characteristics are:

## Purpose

Transfers data from an external debugger to the PE. For example, it is used by a debugger transferring commands and data to a debug target. It is a component of the Debug Communications Channel.

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 DBGDTRRXint is architecturally mapped to AArch64 System register [DBGDTRRX\\_ELO](#).

AArch32 System register DBGDTRRXint is architecturally mapped to External register [DBGDTRRX\\_ELO](#).

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

DBGDTRRXint is a 32-bit register.

## Field descriptions

The DBGDTRRXint 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
Update DTRRX																															

### Bits [31:0]

Update DTRRX.

If RXfull is set to 1, then reads of this register return the last value written to DTRRX and clear RXfull to 0.

For the full behavior of the Debug Communications Channel, see 'The Debug Communication Channel and Instruction Transfer Register' in the ARM ARM, chapter H4.

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

## Accessing the DBGDTRRXint

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
p14, 0, <Rt>, c0, c5, 0	000	000	0000	1110	0101

Data can be stored to memory from this register using STC.

## 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 [DBGDSCRExt](#).UDCCdis==1, read accesses to this register from EL0 are trapped to Undefined mode.
- If [MDSCR\\_EL1](#).TDCC==1, 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](#).TDA==1, Non-secure read 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](#).TDA==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDA==1, read accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

02/05/2017 15:43

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



# DBGDTRTXext, Debug OS Lock Data Transfer Register, Transmit

The DBGDTRTXext characteristics are:

## Purpose

Used for save/restore of [DBGDTRTXint](#). It is a component of the Debug Communication Channel.

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 DBGDTRTXext is architecturally mapped to AArch64 System register [OSDTRTX\\_EL1](#).

## Attributes

DBGDTRTXext is a 32-bit register.

## Field descriptions

The DBGDTRTXext 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
Return DTRTX without side-effect																															

### Bits [31:0]

Return DTRTX without side-effect.

Reads of this register return the value in DTRTX and do not change TXfull.

Writes of this register update the value in DTRTX and do not change TXfull.

For the full behavior of the Debug Communications Channel, see 'The Debug Communication Channel and Instruction Transfer Register' in the ARM ARM, chapter H4.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN on Cold reset. This field is not affected on Warm reset.

## Accessing the DBGDTRTXext

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, c3, 2	000	010	0000	1110	0011

## 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.

ARM deprecates reads and writes of DBGDTRTText through the System register interface when the OS lock is unlocked.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 :

- 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.

02/05/2017 15:43

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

# DBGDTRTXint, Debug Data Transfer Register, Transmit

The DBGDTRTXint characteristics are:

## Purpose

Transfers data from the PE to an external debugger. For example, it is used by a debug target to transfer data to the debugger. It is a component of the Debug Communication Channel.

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 DBGDTRTXint is architecturally mapped to AArch64 System register [DBGDTRTX\\_EL0](#).

AArch32 System register DBGDTRTXint is architecturally mapped to External register [DBGDTRTX\\_EL0](#).

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

DBGDTRTXint is a 32-bit register.

## Field descriptions

The DBGDTRTXint 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
Return DTRTX																															

### Bits [31:0]

Return DTRTX.

If TXfull is set to 0, then writes of this register update the value in DTRTX and set TXfull to 1.

For the full behavior of the Debug Communications Channel, see 'The Debug Communication Channel and Instruction Transfer Register' in the ARM ARM, chapter H4.

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

## Accessing the DBGDTRTXint

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, c5, 0	000	000	0000	1110	0101

Data can be loaded from memory into this register using LDC (immediate) and LDC (literal).

## 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 [DBGDSCRExt](#).UDCCdis==1, write accesses to this register from EL0 are trapped to Undefined mode.
- If [MDSCR\\_EL1](#).TDCC==1, write 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](#).TDA==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](#).TDA==1, Non-secure write accesses to this register from EL0 and EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDA==1, write accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

02/05/2017 15:43

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

# DBGOSDLR, Debug OS Double Lock Register

The DBGOSDLR characteristics are:

## Purpose

Locks out the external debug interface.

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 DBGOSDLR is architecturally mapped to AArch64 System register [OSDLR\\_EL1](#).

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

DBGOSDLR is a 32-bit register.

## Field descriptions

The DBGOSDLR 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	DLK

### Bits [31:1]

Reserved, RES0.

### DLK, bit [0]

OS Double Lock control bit. Possible values are:

DLK	Meaning
0	OS Double Lock unlocked.
1	OS Double Lock locked, if <a href="#">DBGPRCR</a> .CORENPDRQ (Core no powerdown request) bit is set to 0 and the PE is in Non-debug state.

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

## Accessing the DBGOSDLR

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, c3, 4	000	100	0001	1110	0011

## 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<sup>®</sup> 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 :

- If [MDCR\\_EL2](#).TDOSA==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](#).TDOSA==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](#).TDOSA==1, accesses to this register from EL1 and EL2 are trapped to EL3.

# DBGOSECCR, Debug OS Lock Exception Catch Control Register

The DBGOSECCR characteristics are:

## Purpose

Provides a mechanism for an operating system to access the contents of [EDECCR](#) that are otherwise invisible to software, so it can save/restore the contents of [EDECCR](#) over powerdown on behalf of the external debugger.

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 DBGOSECCR is architecturally mapped to AArch64 System register [OSECCR\\_EL1](#).

AArch32 System register DBGOSECCR is architecturally mapped to External register [EDECCR](#).

If OSLSR.OSLK == 0 then DBGOSECCR returns an UNKNOWN value on reads and ignores writes.

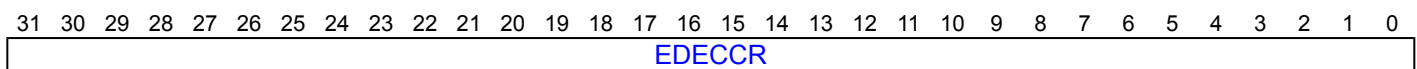
## Attributes

DBGOSECCR is a 32-bit register.

## Field descriptions

The DBGOSECCR bit assignments are:

### When OSLSR.OSLK==1:



### EDECCR, bits [31:0]

Used for save/restore to [EDECCR](#) over powerdown.

Reads or writes to this field are indirect accesses to [EDECCR](#).

## Accessing the DBGOSECCR

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, c6, 2	000	010	0000	1110	0110

## 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.

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.

02/05/2017 15:43

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



# DBGOSLAR, Debug OS Lock Access Register

The DBGOSLAR characteristics are:

## Purpose

Provides a lock for the debug registers. The OS lock also disables some Software debug events.

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 DBGOSLAR is architecturally mapped to AArch64 System register [OSLAR\\_EL1](#).

AArch32 System register DBGOSLAR is architecturally mapped to External register [OSLAR\\_EL1](#).

## Attributes

DBGOSLAR is a 32-bit register.

## Field descriptions

The DBGOSLAR 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
OS Lock Access																															

### Bits [31:0]

OS Lock Access. Writing the value 0xC5ACCE55 to the DBGOSLAR sets the OS lock to 1. Writing any other value sets the OS lock to 0.

Use [DBGOSLSR.OSLK](#) to check the current status of the lock.

## Accessing the DBGOSLAR

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, c0, 4	000	100	0001	1110	0000

## Accessibility

The register is accessible as follows:

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

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 :

- If [MDCR\\_EL2](#).TDOSA==1, Non-secure write accesses to this register from EL1 are trapped to EL2 using AArch64.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

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

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDOSA==1, write accesses to this register from EL1 and EL2 are trapped to EL3 using AArch64.

02/05/2017 15:43

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

# DBGOSLSR, Debug OS Lock Status Register

The DBGOSLSR characteristics are:

## Purpose

Provides status information for the OS lock.

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 DBGOSLSR is architecturally mapped to AArch64 System register [OSLSR\\_EL1](#).

The OS lock status is also visible in the external debug interface through EDPRSR.

This register is in the Cold reset domain. Some or all RW fields of this register have defined reset values. On a Cold reset these apply only if the PE resets into an Exception level that is using AArch32. Otherwise, on a Cold reset RW fields in this register reset to architecturally UNKNOWN values. The register is not affected by a Warm reset.

## Attributes

DBGOSLSR is a 32-bit register.

## Field descriptions

The DBGOSLSR 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	OSLM[1]	nTT	OSLK	OSLM[0]

### Bits [31:4]

Reserved, RES0.

### OSLM[1], bit [3]

See below for description of the OSLM field.

### nTT, bit [2]

Not 32-bit access. This bit is always RAZ. It indicates that a 32-bit access is needed to write the key to the OS Lock Access Register.

### OSLK, bit [1]

OS Lock Status. The possible values are:

OSLK	Meaning
0	OS lock unlocked.
1	OS lock locked.

The OS lock is locked and unlocked by writing to the OS Lock Access Register.

When this register has an architecturally-defined reset value, this field resets to 1.

**OSLM[0], bit [0]**

OS lock model implemented. Identifies the form of OS save and restore mechanism implemented. In ARMv8 these bits are as follows:

OSLM	Meaning
1 0	OS lock implemented. DBGOSLSR not implemented.

All other values are reserved.

**Accessing the DBGOSLSR**

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
p14, 0, <Rt>, c1, c1, 4	000	100	0001	1110	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<sup>®</sup> 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 :

- If [MDCR\\_EL2](#).TDOSA==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 [HDCR](#).TDOSA==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDOSA==1, read accesses to this register from EL1 and EL2 are trapped to EL3.

# DBGPRCR, Debug Power Control Register

The DBGPRCR characteristics are:

## Purpose

Controls behavior of the PE on powerdown request.

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 DBGPRCR is architecturally mapped to AArch64 System register [DBGPRCR\\_ELI](#).

Bit [0] of this register is mapped to [EDPRCR](#).CORENPDRQ, bit [0] of the external view of this register.

The other bits in these registers are not mapped to each other.

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

DBGPRCR is a 32-bit register.

## Field descriptions

The DBGPRCR 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	<a href="#">CORENPDRQ</a>

### Bits [31:1]

Reserved, RES0.

### CORENPDRQ, bit [0]

Core no powerdown request. Requests emulation of powerdown. Possible values of this bit are:

CORENPDRQ	Meaning
0	If the system responds to a powerdown request, it powers down Core power domain.
1	If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain.

Writes to this bit are permitted regardless of the state of the IMPLEMENTATION DEFINED authentication interface. This means that a debugger can request Core no powerdown regardless of whether invasive debug is permitted.

It is IMPLEMENTATION DEFINED whether this bit is reset to the value of [EDPRCR](#).COREPURQ on exit from an IMPLEMENTATION DEFINED software-visible retention state.

## Accessing the DBGPRCR

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, c4, 4	000	100	0001	1110	0100

## 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<sup>®</sup> 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 :

- If [MDCR\\_EL2](#).TDOSA==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](#).TDOSA==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](#).TDOSA==1, accesses to this register from EL1 and EL2 are trapped to EL3.

# DBGVCR, Debug Vector Catch Register

The DBGVCR characteristics are:

## Purpose

Controls Vector Catch debug events.

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 DBGVCR is architecturally mapped to AArch64 System register [DBGVCR32\\_EL2](#).

This register is required in all implementations.

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

DBGVCR is a 32-bit register.

## Field descriptions

The DBGVCR bit assignments are:

### When EL3 implemented and using AArch32:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSF	NSI	0	NSD	NSP	NSS	NSU	0	0	0	0	0	0	0	0	0	MF	MI	0	MD	MP	MS	0	0	SF	SI	0	SD	SP	SS	SU	0

#### NSF, bit [31]

FIQ vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 1C$ .

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

#### NSI, bit [30]

IRQ vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 18$ .

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

#### Bit [29]

Reserved, RES0.

#### NSD, bit [28]

Data Abort vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 10$ .

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

#### **NSP, bit [27]**

Prefetch Abort vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 0C$ .

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

#### **NSS, bit [26]**

Supervisor Call (SVC) vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 08$ .

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

#### **NSU, bit [25]**

Undefined Instruction vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 04$ .

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

#### **Bits [24:16]**

Reserved, RES0.

#### **MF, bit [15]**

FIQ vector catch enable in Monitor mode.

The exception vector offset is  $0 \times 1C$ .

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

#### **MI, bit [14]**

IRQ vector catch enable in Monitor mode.

The exception vector offset is  $0 \times 18$ .

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

#### **Bit [13]**

Reserved, RES0.

#### **MD, bit [12]**

Data Abort vector catch enable in Monitor mode.

The exception vector offset is  $0 \times 10$ .

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



**MP, bit [11]**

Prefetch Abort vector catch enable in Monitor mode.

The exception vector offset is 0x0C.

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

**MS, bit [10]**

Secure Monitor Call (SMC) vector catch enable in Monitor mode.

The exception vector offset is 0x08.

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

**Bits [9:8]**

Reserved, RES0.

**SF, bit [7]**

FIQ vector catch enable in Secure state.

The exception vector offset is 0x1C.

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

**SI, bit [6]**

IRQ vector catch enable in Secure state.

The exception vector offset is 0x18.

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

**Bit [5]**

Reserved, RES0.

**SD, bit [4]**

Data Abort vector catch enable in Secure state.

The exception vector offset is 0x10.

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

**SP, bit [3]**

Prefetch Abort vector catch enable in Secure state.

The exception vector offset is 0x0C.

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

**SS, bit [2]**

Supervisor Call (SVC) vector catch enable in Secure state.

The exception vector offset is 0x08.

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

### SU, bit [1]

Undefined Instruction vector catch enable in Secure state.

The exception vector offset is  $0 \times 04$ .

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

### Bit [0]

Reserved, RES0.

## When EL3 implemented and using AArch64:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSF	NSI	0	NSD	NSP	NSS	NSU	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SF	SI	0	SD	SP	SS	SU	0

### NSF, bit [31]

FIQ vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 1C$ .

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

### NSI, bit [30]

IRQ vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 18$ .

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

### Bit [29]

Reserved, RES0.

### NSD, bit [28]

Data Abort vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 10$ .

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

### NSP, bit [27]

Prefetch Abort vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 0C$ .

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

### NSS, bit [26]

Supervisor Call (SVC) vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 08$ .

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

**NSU, bit [25]**

Undefined Instruction vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 04$ .

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

**Bits [24:8]**

Reserved, RES0.

**SF, bit [7]**

FIQ vector catch enable in Secure state.

The exception vector offset is  $0 \times 1C$ .

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

**SI, bit [6]**

IRQ vector catch enable in Secure state.

The exception vector offset is  $0 \times 18$ .

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

**Bit [5]**

Reserved, RES0.

**SD, bit [4]**

Data Abort vector catch enable in Secure state.

The exception vector offset is  $0 \times 10$ .

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

**SP, bit [3]**

Prefetch Abort vector catch enable in Secure state.

The exception vector offset is  $0 \times 0C$ .

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

**SS, bit [2]**

Supervisor Call (SVC) vector catch enable in Secure state.

The exception vector offset is  $0 \times 08$ .

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

**SU, bit [1]**

Undefined Instruction vector catch enable in Secure state.

The exception vector offset is  $0 \times 04$ .

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

### Bit [0]

Reserved, RES0.

## When EL3 not 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	0	0	0	0	0	0	0	0	F	I	0	D	P	S	U	0

### Bits [31:8]

Reserved, RES0.

### F, bit [7]

FIQ vector catch enable.

The exception vector offset is  $0 \times 1C$ .

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

### I, bit [6]

IRQ vector catch enable.

The exception vector offset is  $0 \times 18$ .

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

### Bit [5]

Reserved, RES0.

### D, bit [4]

Data Abort vector catch enable.

The exception vector offset is  $0 \times 10$ .

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

### P, bit [3]

Prefetch Abort vector catch enable.

The exception vector offset  $0 \times 0C$ .

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

### S, bit [2]

Supervisor Call (SVC) vector catch enable.

The exception vector offset is  $0 \times 08$ .

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

**U, bit [1]**

Undefined Instruction vector catch enable.

The exception vector offset is  $0 \times 04$ .

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

**Bit [0]**

Reserved, RES0.

**Accessing the DBGVCR**

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, c7, 0	000	000	0000	1110	0111

**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<sup>®</sup> 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 :

- 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.

# 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	MASK					0	0	0	WT	LBN			SSC		HMC	BAS							LSC	PAC	E				

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 <a href="#">DBGWVR&lt;n&gt;</a>
xxxxxxx1x	Match byte at <a href="#">DBGWVR&lt;n&gt;+1</a>
xxxxx1xx	Match byte at <a href="#">DBGWVR&lt;n&gt;+2</a>
xxxx1xxx	Match byte at <a href="#">DBGWVR&lt;n&gt;+3</a>

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

BAS	Description, if <a href="#">DBGWVR&lt;n&gt;</a> [2] == 0
xxx1xxxx	Match byte at <a href="#">DBGWVR&lt;n&gt;+4</a>
xx1xxxxx	Match byte at <a href="#">DBGWVR&lt;n&gt;+5</a>
x1xxxxxx	Match byte at <a href="#">DBGWVR&lt;n&gt;+6</a>
1xxxxxxx	Match byte at <a href="#">DBGWVR&lt;n&gt;+7</a>

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, accesses to this register from PL1 and PL2 are trapped to Debug 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.

02/05/2017 15:43

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

# DBGWFAR, Debug Watchpoint Fault Address Register

The DBGWFAR characteristics are:

## Purpose

Previously returned information about the address of the instruction that accessed a watchpointed address. Is now deprecated and RES0.

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.

## Attributes

DBGWFAR is a 32-bit register.

## Field descriptions

The DBGWFAR 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	0

### Bits [31:0]

Reserved, RES0.

## Accessing the DBGWFAR

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, c6, 0	000	000	0000	1110	0110

## 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.

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.

02/05/2017 15:43

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

# 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, accesses to this register from PL1 and PL2 are trapped to Debug 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.

# DFAR, Data Fault Address Register

The DFAR characteristics are:

## Purpose

Holds the virtual address of the faulting address that caused a synchronous Data Abort exception.

This register is part of the Exception and fault handling registers functional group.

## Configuration

AArch32 System register DFAR (NS) is architecturally mapped to AArch64 System register [FAR\\_EL1\[31:0\]](#).

AArch32 System register DFAR (S) is architecturally mapped to AArch32 System register [HDFAR](#) when EL2 is implemented.

AArch32 System register DFAR (S) is architecturally mapped to AArch64 System register [FAR\\_EL2\[31:0\]](#) when EL2 is implemented.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

DFAR is a 32-bit register.

## Field descriptions

The DFAR 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 of faulting address of synchronous Data Abort exception																															

### Bits [31:0]

VA of faulting address of synchronous Data Abort exception.

## Accessing the DFAR

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>, c6, c0, 0	000	000	0110	1111	0000

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	

EL3 using AArch32	x	x	0	-	n/a	n/a	RW	DFAR_s
EL3 not implemented	x	x	0	-	RW	n/a	n/a	DFAR
EL3 not implemented	x	0	1	-	RW	RW	n/a	DFAR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	DFAR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	DFAR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	DFAR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	DFAR
EL3 using AArch32	x	0	1	-	RW	RW	RW	DFAR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	DFAR_ns

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<sup>®</sup> 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.T6==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.T6==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.T6==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# DFSR, Data Fault Status Register

The DFSR characteristics are:

## Purpose

Holds status information about the last data fault.

This register is part of the Exception and fault handling registers functional group.

## Configuration

AArch32 System register DFSR is architecturally mapped to AArch64 System register [ESR\\_EL1](#).

The current translation table format determines which format of the register is used.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

DFSR is a 32-bit register.

## Field descriptions

The DFSR bit assignments are:

### When TTBCR.EAE==0:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FnV	AET	CM	Ext	WnR	FS[4]	LPAE	0		Domain					FS[3:0]		

### Bits [31:17]

Reserved, RES0.

### FnV, bit [16]

FAR not Valid, for a Synchronous external abort other than a Synchronous external abort on a translation table walk.

FnV	Meaning
0	<a href="#">DFAR</a> is valid.
1	<a href="#">DFAR</a> is not valid, and holds an UNKNOWN value.

This field is only valid for a Synchronous external abort other than a Synchronous external abort on a translation table walk. It is RES0 for all other Data Abort exceptions.

### AET, bits [15:14]

Asynchronous Error Type. When the RAS Extension is implemented, this field describes the state of the PE after taking an asynchronous Data Abort exception. Possible values are:

AET	Meaning
00	Uncontainable error (UC) or uncategorized.
01	Unrecoverable error (UEU).
10	Restartable error (UEO) or Corrected error (CE).
11	Recoverable error (UER).



When the RAS Extension is not implemented, or on a synchronous Data Abort, this field is RES0.

#### Note

ARMv8.2 requires the implementation of the RAS Extension.

In the event of multiple errors taken as a single SError interrupt exception, the overall state of the PE is reported.

#### Note

Software can use this information to determine what recovery might be possible. The recovery software must also examine any implemented fault records to determine the location and extent of the error.

### CM, bit [13]

Cache maintenance fault. For synchronous faults, this bit indicates whether a cache maintenance instruction generated the fault. The possible values of this bit are:

CM	Meaning
0	Abort not caused by execution of a cache maintenance instruction.
1	Abort caused by execution of a cache maintenance instruction.

On a synchronous Data Abort on a translation table walk, this bit is UNKNOWN.

On an asynchronous fault, this bit is UNKNOWN.

### ExT, bit [12]

External abort type. This bit can be used to provide an IMPLEMENTATION DEFINED classification of external aborts.

In an implementation that does not provide any classification of external aborts, this bit is RES0.

For aborts other than external aborts this bit always returns 0.

### WnR, bit [11]

Write not Read bit. Indicates whether the abort was caused by a write or a read instruction. The possible values of this bit are:

WnR	Meaning
0	Abort caused by a read instruction.
1	Abort caused by a write instruction.

For faults on the cache maintenance and address translation System instructions in the (coproc==1111) encoding space this bit always returns a value of 1.

### FS[4], bit [10]

See FS[3:0], bits [3:0] for description of the FS field.

### LPAE, bit [9]

On taking a Data Abort exception, this bit is set as follows:

LPAE	Meaning
0	Using the Short-descriptor translation table formats.
1	Using the Long-descriptor translation table formats.

Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.

## Bit [8]

Reserved, RES0.

## Domain, bits [7:4]

The domain of the fault address.

ARM deprecates any use of this field, see 'The Domain field in the DFSR' in the ARMv8 ARM.

This field is UNKNOWN for certain faults where the DFSR is updated and reported using the Short-descriptor FSR encodings, see 'Validity of Domain field on faults that update the DFSR when using the Short-descriptor encodings' in the ARMv8 ARM.

## FS[3:0], bits [3:0]

Fault status bits. Interpreted with bit [10]. Possible values of FS[4:0] are:

FS	Meaning
00001	Alignment fault
00010	Debug exception
00011	Access flag fault, level 1
00100	Fault on instruction cache maintenance
00101	Translation fault, level 1
00110	Access flag fault, level 2
00111	Translation fault, level 2
01000	Synchronous external abort, not on translation table walk
01001	Domain fault, level 1
01011	Domain fault, level 2
01100	Synchronous external abort, on translation table walk, level 1
01101	Permission fault, level 1
01110	Synchronous external abort, on translation table walk, level 2
01111	Permission fault, level 2
10000	TLB conflict abort
10100	IMPLEMENTATION DEFINED fault (Lockdown fault)
10101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive access fault)
10110	SError interrupt
11000	SError interrupt, from a parity or ECC error on memory access
11001	Synchronous parity or ECC error on memory access, not on translation table walk
11100	Synchronous parity or ECC error on translation table walk, level 1
11110	Synchronous parity or ECC error on translation table walk, level 2

All other values are reserved.

When the RAS Extension is implemented, 11000, 11001, 11100, and 11110, are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults on a Short-descriptor translation table lookup' in the ARMv8 ARM.

## When TTBCR.EAE==1:

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

## Bits [31:17]

Reserved, RES0.

## FnV, bit [16]

FAR not Valid, for a Synchronous external abort other than a Synchronous external abort on a translation table walk.

<b>FnV</b>	<b>Meaning</b>
0	<a href="#">DFAR</a> is valid.
1	<a href="#">DFAR</a> is not valid, and holds an UNKNOWN value.

This field is only valid for a Synchronous external abort other than a Synchronous external abort on a translation table walk. It is RES0 for all other Data Abort exceptions.

### AET, bits [15:14]

Asynchronous Error Type. When the RAS Extension is implemented, this field describes the state of the PE after taking an asynchronous Data Abort exception. Possible values are:

<b>AET</b>	<b>Meaning</b>
00	Uncontainable error (UC) or uncategorized.
01	Unrecoverable error (UEU).
10	Restartable error (UEO) or Corrected error (CE).
11	Recoverable error (UER).

When the RAS Extension is not implemented, or on a synchronous Data Abort, this field is RES0.

#### Note

ARMv8.2 requires the implementation of the RAS Extension.

In the event of multiple errors taken as a single SError interrupt exception, the overall state of the PE is reported.

#### Note

Software can use this information to determine what recovery might be possible. The recovery software must also examine any implemented fault records to determine the location and extent of the error.

### CM, bit [13]

Cache maintenance fault. For synchronous faults, this bit indicates whether a cache maintenance instruction generated the fault. The possible values of this bit are:

<b>CM</b>	<b>Meaning</b>
0	Abort not caused by execution of a cache maintenance instruction.
1	Abort caused by execution of a cache maintenance instruction.

On a synchronous Data Abort on a translation table walk, this bit is UNKNOWN.

On an asynchronous fault, this bit is UNKNOWN.

### ExT, bit [12]

External abort type. This bit can be used to provide an IMPLEMENTATION DEFINED classification of external aborts.

In an implementation that does not provide any classification of external aborts, this bit is RES0.

For aborts other than external aborts this bit always returns 0.

### WnR, bit [11]

Write not Read bit. Indicates whether the abort was caused by a write or a read instruction. The possible values of this bit are:

<b>WnR</b>	<b>Meaning</b>
0	Abort caused by a read instruction.
1	Abort caused by a write instruction.

For faults on the cache maintenance and address translation System instructions in the (coproc==1111) encoding space this bit always returns a value of 1.

**Bit [10]**

Reserved, RES0.

**LPAE, bit [9]**

On taking a Data Abort exception, this bit is set as follows:

LPAE	Meaning
0	Using the Short-descriptor translation table formats.
1	Using the Long-descriptor translation table formats.

Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.

**Bits [8:6]**

Reserved, RES0.

**STATUS, bits [5:0]**

Fault status bits. Possible values of this field are:

STATUS	Meaning
000000	Address size fault in <a href="#">TTBR0</a> or <a href="#">TTBR1</a>
000001	Address size fault, level 1
000010	Address size fault, level 2
000011	Address size fault, level 3
000101	Translation fault, level 1
000110	Translation fault, level 2
000111	Translation fault, level 3
001001	Access flag fault, level 1
001010	Access flag fault, level 2
001011	Access flag fault, level 3
001101	Permission fault, level 1
001110	Permission fault, level 2
001111	Permission fault, level 3
010000	Synchronous external abort, not on translation table walk
010001	SError interrupt
010101	Synchronous external abort, on translation table walk, level 1
010110	Synchronous external abort, on translation table walk, level 2
010111	Synchronous external abort, on translation table walk, level 3
011000	Synchronous parity or ECC error on memory access, not on translation table walk
011001	SError interrupt, from a parity or ECC error on memory access
011101	Synchronous parity or ECC error on memory access on translation table walk, level 1
011110	Synchronous parity or ECC error on memory access on translation table walk, level 2
011111	Synchronous parity or ECC error on memory access on translation table walk, level 3
100001	Alignment fault
100010	Debug exception
110000	TLB conflict abort
110100	IMPLEMENTATION DEFINED fault (Lockdown fault)
110101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive access fault)

All other values are reserved.

When the RAS Extension is implemented, 011000, 011001, 011101, 011110, and 011111, are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults on a Long-descriptor translation table lookup' in the ARMv8 ARM.

## Accessing the DFSR

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>, c5, c0, 0	000	000	0101	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	DFSR
EL3 not implemented	x	0	1	-	RW	RW	n/a	DFSR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	DFSR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	DFSR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	DFSR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	DFSR
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	DFSR_s
EL3 using AArch32	x	0	1	-	RW	RW	RW	DFSR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	DFSR_ns

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.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.T5==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.T5==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.T5==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.



# DLR, Debug Link Register

The DLR characteristics are:

## Purpose

In Debug state, holds the address to restart from.

This register is part of:

- The Debug registers functional group.
- The Special-purpose registers functional group.

## Configuration

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

AArch32 System register DLR is architecturally mapped to AArch64 System register [DLR\\_EL0\[31:0\]](#).

## Attributes

DLR is a 32-bit register.

## Field descriptions

The DLR 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
Restart address																															

### Bits [31:0]

Restart address.

## Accessing the DLR

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, 3, <Rt>, c4, c5, 1	011	001	0100	1111	0101

## Accessibility

The register is accessible as follows:

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

# DLR, Debug Link Register

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.

Access to this register is from Debug state only. During normal execution this register is unallocated.

02/05/2017 15:43

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



# DSPSR, Debug Saved Program Status Register

The DSPSR characteristics are:

## Purpose

Holds the saved process state on entry to Debug state.

This register is part of:

- The Debug registers functional group.
- The Special-purpose registers functional group.

## Configuration

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

AArch32 System register DSPSR is architecturally mapped to AArch64 System register [DSPSR\\_ELO](#).

## Attributes

DSPSR is a 32-bit register.

## Field descriptions

The DSPSR bit assignments are:

### When entering Debug state from AArch32 and exiting Debug state to AArch32:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	Z	C	V	Q	IT[1:0]	J	0	PAN	SS	IL		GE		IT[7:2]		E	A	I	F	T	M[4]		M[3:0]								

#### N, bit [31]

Set to the value of [CPSR.N](#) on entering Debug state, and copied to [CPSR.N](#) on exiting Debug state.

#### Z, bit [30]

Set to the value of [CPSR.Z](#) on entering Debug state, and copied to [CPSR.Z](#) on exiting Debug state.

#### C, bit [29]

Set to the value of [CPSR.C](#) on entering Debug state, and copied to [CPSR.C](#) on exiting Debug state.

#### V, bit [28]

Set to the value of [CPSR.V](#) on entering Debug state, and copied to [CPSR.V](#) on exiting Debug state.

#### Q, bit [27]

Set to the value of [CPSR.Q](#) on entering Debug state, and copied to [CPSR.Q](#) on exiting Debug state.

#### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR](#).PAN on entering Debug state, and copied to [CPSR](#).PAN on exiting Debug state.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**SS, bit [21]**

Software step. Shows the value of PSTATE.SS immediately before Debug state was entered.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before Debug state was entered.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

<b>E</b>	<b>Meaning</b>
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTLR.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

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

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

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

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the Debug state entry was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that Debug state was entered from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that Debug state was entered from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor (only valid in Secure state, if EL3 is implemented and can use AArch32)
0b0111	Abort
0b1010	Hyp
0b1011	Undefined
0b1111	System

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

## Accessing the DSPSR

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, 3, <Rt>, c4, c5, 0	011	000	0100	1111	0101

## 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.

Access to this register is from Debug state only. During normal execution this register is unallocated.

02/05/2017 15:43

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

# ELR\_hyp, Exception Link Register (Hyp mode)

The ELR\_hyp characteristics are:

## Purpose

When taking an exception to Hyp mode, holds the address to return to.

This register is part of the Special-purpose registers functional group.

## Configuration

AArch32 System register ELR\_hyp is architecturally mapped to AArch64 System register [ELR\\_EL2](#).

On a reset into an Exception level that is using AArch32 ELR\_hyp is UNKNOWN.

## Attributes

ELR\_hyp is a 32-bit register.

## Field descriptions

The ELR\_hyp 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
																Return address															

### Bits [31:0]

Return address.

## Accessing the ELR\_hyp

This register can be read using MRS (banked register) with the following syntax:

```
MRS <Rd>, <banked_reg>
```

This register can be written using MSR (banked register) with the following syntax:

```
MSR <banked_reg>, <Rd>
```

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

<banked_reg>	R	M	M1
ELR_hyp	0	1	1110

## 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.

02/05/2017 15:43

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

# FCSEIDR, FCSE Process ID register

The FCSEIDR characteristics are:

## Purpose

Identifies whether the Fast Context Switch Extension (FCSE) is implemented.

In ARMv8, the FCSE is not implemented, so this register is RAZ/WI. Software can access this register to determine that the implementation does not include the FCSE.

This register is part of the Legacy feature registers functional group.

## Configuration

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

## Attributes

FCSEIDR is a 32-bit register.

## Field descriptions

The FCSEIDR 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	0

### Bits [31:0]

Reserved, RAZ/WI. Hardware must implement this as RAZ/WI. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

## Accessing the FCSEIDR

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>, c13, c0, 0	000	000	1101	1111	0000

## 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<sup>®</sup> 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](#).T13==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](#).T13==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](#).T13==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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



# FPEXC, Floating-Point Exception Control register

The FPEXC characteristics are:

## Purpose

Provides a global enable for the implemented Advanced SIMD and floating-point functionality, and reports floating-point status information.

This register is part of the Floating-point registers functional group.

## Configuration

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

AArch32 System register FPEXC is architecturally mapped to AArch64 System register [FPEXC32\\_EL2](#).

Implemented only if the implementation includes the Advanced SIMD and floating-point functionality.

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. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

FPEXC is a 32-bit register.

## Field descriptions

The FPEXC 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
EX	EN	DEX	FP	2V	VV	TF	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VEC	ITR	IDF	0	0	IXF	UFF	OFF	DZF	IOF	

### EX, bit [31]

Exception bit. In ARMv8, this bit is RAZ/WI.

### EN, bit [30]

Enables access to the Advanced SIMD and floating-point functionality from all Exception levels, except that setting this field to 0 does not disable the following:

- VMSR accesses to the [FPEXC](#) or [FPSID](#).
- VMRS accesses from the [FPEXC](#), [FPSID](#), [MVFR0](#), [MVFR1](#), or [MVFR2](#).

EN	Meaning
0	Accesses to the <a href="#">FPSCR</a> , and any of the SIMD and floating-point registers Q0-Q15, including their views as D0-D31 registers or S0-S31 registers, are UNDEFINED at all Exception levels.
1	This control permits access to the Advanced SIMD and floating-point functionality at all Exception levels.

Execution of floating-point and Advanced SIMD instructions in AArch32 state can be disabled or trapped by the following controls:

- [CPACR](#).cp10, or, if executing at EL0, [CPACR\\_EL1](#).FPEN.
- FPEXC.EN.
- If executing in Non-secure state:
  - [HCPTR](#).TCP10, or if EL2 is using AArch64, [CPTR\\_EL2](#).TFP.
  - [NSACR](#).cp10, or if EL3 is using AArch64, [CPTR\\_EL3](#).TFP.
- For Advanced SIMD instructions only:
  - CPACR.ASEDIS.

- If executing in Non-secure state, [HCPTR](#).TASE and [NSACR](#).NSTRCDIS.

See the descriptions of the controls for more information.

---

#### Note

When executing at EL0 using AArch32:

- If EL1 is using AArch64 then behavior is as if the value of FPEXC.EN is 1.
  - In Non-secure state, if EL2 is using AArch64 and the value of [HCR\\_EL2](#).{RW, TGE} is {1, 1} then behavior is as if the value of FPEXC.EN is 1.
  - In Non-secure state, if EL2 is using AArch64 and the value of [HCR\\_EL2](#).{RW, TGE} is {0, 1} then it is IMPLEMENTATION DEFINED whether the behavior is:
    - As if the value of FPEXC.EN is 1.
    - Determined by the value of FPEXC.EN, as described in this field description. However, ARM deprecates using the value of FPEXC.EN to determine behavior.
- 

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

#### DEX, bit [29]

Defined synchronous exception on floating-point execution.

This field identifies whether a synchronous exception generated by the attempted execution of an instruction was generated by an unallocated encoding. The instruction must be in the encoding space that is identified by the pseudocode function `ExecutingCP10or11Instr()` returning TRUE. This field also indicates whether the FPEXC.TFV field is valid.

The meaning of this bit is:

DEX	Meaning
0	The exception was generated by the attempted execution of an unallocated instruction in the encoding space that is identified by the pseudocode function <code>ExecutingCP10or11Instr()</code> . If FPEXC.TFV is RW then it is invalid and UNKNOWN. If FPEXC.{IDF, IXF, UFF, OFF, DZF, IOF} are RW then they are invalid and UNKNOWN.
1	The exception was generated during the execution of an unallocated encoding. FPEXC.TFV is valid and indicates the cause of the exception.

On an exception that sets this bit to 1 the exception-handling routine must clear this bit to 0.

On an implementation that both does not support trapping of floating-point exceptions and implements the [FPSCR](#).{Stride, Len} fields as RAZ, 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 a value that is architecturally UNKNOWN.

#### FP2V, bit [28]

FPINST2 instruction valid bit. In ARMv8, this bit is RES0.

#### VV, bit [27]

VECITR valid bit. In ARMv8, this bit is RES0.

#### TFV, bit [26]

Trapped Fault Valid bit. Valid only when the value of FPEXC.DEX is 1. When valid, it indicates the cause of the exception and therefore whether the FPEXC.{IDF, IXF, UFF, OFF, DZF, IOF} bits are valid.

TFV	Meaning
0	The exception was caused by the execution of a floating-point VABS, VADD, VDIV, VFMA, VFMS, VFNMA, VFNMS, VMLA, VMLS, VMOV, VMUL, VNEG, VNMLA, VNMLS, VNMUL, VSQRT, or VSUB instruction when one or both of <a href="#">FPSCR</a> .{Stride, Len} was non-zero. If the FPEXC.{IDF, IXF, UFF, OFF, DZF, IOF} bits are RW then they are invalid and UNKNOWN.
1	FPEXC.{IDF, IXF, UFF, OFF, DZF, IOF} indicate the presence of trapped floating-point exceptions that had occurred at the time of the exception. Bits are set for all trapped exceptions that had occurred at the time of the exception.

This bit returns a status value and ignores writes.

When the value of FPEXC.DEX is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions this bit is RAZ/WI.

On an implementation that supports the trapping of floating-point exceptions and implements [FPSCR](#).{Stride, Len} as RAZ, this bit is RAO/WI.

## Bits [25:11]

Reserved, RES0.

## VECITR, bits [10:8]

Vector iteration count. In ARMv8, this field is RES1.

## IDF, bit [7]

Input Denormal trapped exception bit. Valid only when the value of FPEXC.TFV is 1. When valid, it indicates whether an Input Denormal exception occurred while [FPSCR](#).IDE was 1:

IDF	Meaning
0	Input denormal exception has not occurred.
1	Input denormal exception has occurred.

Input Denormal exceptions can occur only when [FPSCR](#).FZ is 1.

This bit must be cleared to 0 by the exception-handling routine.

When the value of FPEXC.TFV is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions 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.

## Bits [6:5]

Reserved, RES0.

## IXF, bit [4]

Inexact trapped exception bit. Valid only when the value of FPEXC.TFV is 1. When valid, it indicates whether an Inexact exception occurred while [FPSCR](#).IXE was 1:

IXF	Meaning
0	Inexact exception has not occurred.
1	Inexact exception has occurred.

This bit must be cleared to 0 by the exception-handling routine.

When the value of FPEXC.TFV is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions 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.

### UFF, bit [3]

Underflow trapped exception bit. Valid only when the value of FPEXC.TFV is 1. When valid, it indicates whether an Underflow exception occurred while [FPSCR.UFE](#) was 1:

UFF	Meaning
0	Underflow exception has not occurred.
1	Underflow exception has occurred.

Underflow trapped exceptions can occur only when [FPSCR.FZ](#) is 0.

This bit must be cleared to 0 by the exception-handling routine.

When the value of FPEXC.TFV is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions 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.

### OFF, bit [2]

Overflow trapped exception bit. Valid only when the value of FPEXC.TFV is 1. When valid, it indicates whether an Overflow exception occurred while [FPSCR.OFE](#) was 1:

OFF	Meaning
0	Overflow exception has not occurred.
1	Overflow exception has occurred.

This bit must be cleared to 0 by the exception-handling routine.

When the value of FPEXC.TFV is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions 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.

### DZF, bit [1]

Divide by Zero trapped exception bit. Valid only when the value of FPEXC.TFV is 1. When valid, it indicates whether a Divide by Zero exception occurred while [FPSCR.DZE](#) was 1:

DZF	Meaning
0	Divide by Zero exception has not occurred.
1	Divide by Zero exception has occurred.

This bit must be cleared to 0 by the exception-handling routine.

When the value of FPEXC.TFV is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions 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.

### IOF, bit [0]

Invalid Operation trapped exception bit. Valid only when the value of FPEXC.TFV is 1. When valid, it indicates whether an Invalid Operation exception occurred while [FPSCR.IOE](#) was 1:

IOF	Meaning
0	Invalid Operation exception has not occurred.
1	Invalid Operation exception has occurred.

This bit must be cleared to 0 by the exception-handling routine.

When the value of FPEXC.TFV is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions 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.

## Accessing the FPEXC

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

```
VMRS <Rt>, <spec_reg>
```

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

```
VMSR <spec_reg>, <Rt>
```

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

<spec_reg>	reg
FPEXC	1000

## 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.

When HCR\_EL2.E2H==0 :

- If [CPACR](#).cp10==00, accesses to this register from PL1 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [CPTR\\_EL2](#).TFP==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 [CPTR\\_EL2](#).FPEN==00, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [CPTR\\_EL2](#).FPEN==10, 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 [HCPTR](#).TCP10==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

- If [HCPTR.TCP10](#)==1, Non-secure accesses to this register from EL2 are UNDEFINED.

When EL3 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [NSACR.cp10](#)==0, Non-secure accesses to this register from EL1 and EL2 are UNDEFINED.

When EL3 is implemented and is using AArch64 :

- If [CPTR\\_EL3.TFP](#)==1, accesses to this register from EL1 and EL2 are trapped to EL3.

02/05/2017 15:43

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

# FPSCR, Floating-Point Status and Control Register

The FPSCR characteristics are:

## Purpose

Provides floating-point system status information and control.

This register is part of:

- The Special-purpose registers functional group.
- The Floating-point registers functional group.

## Configuration

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

The named fields in this register map to the equivalent fields in the AArch64 [FPCR](#) and [FPSR](#).

It is IMPLEMENTATION DEFINED whether the Len and Stride fields can be programmed to non-zero values, which will cause some AArch32 floating-point instruction encodings to be UNDEFINED, or whether these fields are RAZ.

Implemented only if the implementation includes the Advanced SIMD and floating-point functionality.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

FPSCR is a 32-bit register.

## Field descriptions

The FPSCR 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
N	Z	C	V	QC	AHP	DN	FZ	RMode	Stride	FZ16	Len	IDE	0	0	IXE	UFE	OFD	DZE	IOE	IDC	0	0	IXC	UFC	OFD	DZC	IOC				

### N, bit [31]

Negative condition flag. This is updated by floating-point comparison operations.

### Z, bit [30]

Zero condition flag. This is updated by floating-point comparison operations.

### C, bit [29]

Carry condition flag. This is updated by floating-point comparison operations.

### V, bit [28]

Overflow condition flag. This is updated by floating-point comparison operations.

### QC, bit [27]

Cumulative saturation bit, Advanced SIMD only. This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated since 0 was last written to this bit.

**AHP, bit [26]**

Alternative half-precision control bit:

AHP	Meaning
0	IEEE half-precision format selected.
1	Alternative half-precision format selected.

This bit is only used for conversions between half-precision floating-point and other floating-point formats.

The data-processing instructions added as part of the ARMv8.2-FP16 extension always use the IEEE half-precision format, and ignore the value of this bit.

**DN, bit [25]**

Default NaN mode control bit:

DN	Meaning
0	NaN operands propagate through to the output of a floating-point operation.
1	Any operation involving one or more NaNs returns the Default NaN.

The value of this bit only controls scalar floating-point arithmetic. Advanced SIMD arithmetic always uses the Default NaN setting, regardless of the value of the DN bit.

**FZ, bit [24]**

Flush-to-zero mode control bit:

FZ	Meaning
0	Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.
1	Flush-to-zero mode enabled.

The value of this bit only controls scalar floating-point arithmetic. Advanced SIMD arithmetic always uses the Flush-to-zero setting, regardless of the value of the FZ bit.

This bit has no effect on half-precision calculations.

**RMode, bits [23:22]**

Rounding Mode control field. The encoding of this field is:

RMode	Meaning
00	Round to Nearest (RN) mode
01	Round towards Plus Infinity (RP) mode
10	Round towards Minus Infinity (RM) mode
11	Round towards Zero (RZ) mode.

The specified rounding mode is used by almost all scalar floating-point instructions. Advanced SIMD arithmetic always uses the Round to Nearest setting, regardless of the value of the RMode bits.

**Stride, bits [21:20]**

It is IMPLEMENTATION DEFINED whether this field is RW or RAZ.

If this field is RW and is set to a value other than zero, some floating-point instruction encodings are UNDEFINED. The instruction pseudocode identifies these instructions.

ARM strongly recommends that software never sets this field to a value other than zero.

The value of this field is ignored when processing Advanced SIMD instructions.



**FZ16, bit [19]****In ARMv8.2:**

When ARMv8.2-FP16 is implemented, flush-to-zero mode control bit on half-precision data-processing instructions:

<b>FZ16</b>	<b>Meaning</b>
0	Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.
1	Flush-to-zero mode enabled.

The value of this bit applies to both scalar and Advanced SIMD floating-point half-precision calculations.

When ARMv8.2-FP16 is not implemented, this bit is RES0.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**Len, bits [18:16]**

It is IMPLEMENTATION DEFINED whether this field is RW or RAZ.

If this field is RW and is set to a value other than zero, some floating-point instruction encodings are UNDEFINED. The instruction pseudocode identifies these instructions.

ARM strongly recommends that software never sets this field to a value other than zero.

The value of this field is ignored when processing Advanced SIMD instructions.

**IDE, bit [15]**

Input Denormal floating-point exception trap enable. Possible values are:

<b>IDE</b>	<b>Meaning</b>
0	Untrapped exception handling selected. If the floating-point exception occurs then the IDC bit is set to 1.
1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the IDC bit. The trap handling software can decide whether to set the IDC bit to 1.

This bit is RW only if the implementation supports the trapping of floating-point exceptions. In an implementation that does not support floating-point exception trapping, this bit is RES0.

When this bit is RW, it applies only to floating-point operations. Advanced SIMD operations always use untrapped floating-point exception handling in AArch32 state.

**Bits [14:13]**

Reserved, RES0.

**IXE, bit [12]**

Inexact floating-point exception trap enable. Possible values are:

<b>IXE</b>	<b>Meaning</b>
0	Untrapped exception handling selected. If the floating-point exception occurs then the IXC bit is set to 1.
1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the IXC bit. The trap handling software can decide whether to set the IXC bit to 1.

This bit is RW only if the implementation supports the trapping of floating-point exceptions. In an implementation that does not support floating-point exception trapping, this bit is RES0.

When this bit is RW, it applies only to floating-point operations. Advanced SIMD operations always use untrapped floating-point exception handling in AArch32 state.

### UFE, bit [11]

Underflow floating-point exception trap enable. Possible values are:

UFE	Meaning
0	Untrapped exception handling selected. If the floating-point exception occurs then the UFC bit is set to 1.
1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the UFC bit. The trap handling software can decide whether to set the UFC bit to 1.

This bit is RW only if the implementation supports the trapping of floating-point exceptions. In an implementation that does not support floating-point exception trapping, this bit is RES0.

When this bit is RW, it applies only to floating-point operations. Advanced SIMD operations always use untrapped floating-point exception handling in AArch32 state.

### OFE, bit [10]

Overflow floating-point exception trap enable. Possible values are:

OFE	Meaning
0	Untrapped exception handling selected. If the floating-point exception occurs then the OFC bit is set to 1.
1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the OFC bit. The trap handling software can decide whether to set the OFC bit to 1.

This bit is RW only if the implementation supports the trapping of floating-point exceptions. In an implementation that does not support floating-point exception trapping, this bit is RES0.

When this bit is RW, it applies only to floating-point operations. Advanced SIMD operations always use untrapped floating-point exception handling in AArch32 state.

### DZE, bit [9]

Divide by Zero floating-point exception trap enable. Possible values are:

DZE	Meaning
0	Untrapped exception handling selected. If the floating-point exception occurs then the DZC bit is set to 1.
1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the DZC bit. The trap handling software can decide whether to set the DZC bit to 1.

This bit is RW only if the implementation supports the trapping of floating-point exceptions. In an implementation that does not support floating-point exception trapping, this bit is RES0.

When this bit is RW, it applies only to floating-point operations. Advanced SIMD operations always use untrapped floating-point exception handling in AArch32 state.

### IOE, bit [8]

Invalid Operation floating-point exception trap enable. Possible values are:

IOE	Meaning
0	Untrapped exception handling selected. If the floating-point exception occurs then the IOC bit is set to 1.
1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the IOC bit. The trap handling software can decide whether to set the IOC bit to 1.

This bit is RW only if the implementation supports the trapping of floating-point exceptions. In an implementation that does not support floating-point exception trapping, this bit is RES0.

When this bit is RW, it applies only to floating-point operations. Advanced SIMD operations always use untrapped floating-point exception handling in AArch32 state.

#### **IDC, bit [7]**

Input Denormal cumulative floating-point exception bit. This bit is set to 1 to indicate that the Input Denormal floating-point exception has occurred since 0 was last written to this bit.

How VFP instructions update this bit depends on the value of the IDE bit.

Advanced SIMD instructions set this bit if the Input Denormal floating-point exception occurs in one or more of the floating-point calculations performed by the instruction, regardless of the value of the IDE bit.

#### **Bits [6:5]**

Reserved, RES0.

#### **IXC, bit [4]**

Inexact cumulative floating-point exception bit. This bit is set to 1 to indicate that the Inexact floating-point exception has occurred since 0 was last written to this bit.

How VFP instructions update this bit depends on the value of the IXE bit.

Advanced SIMD instructions set this bit if the Inexact floating-point exception occurs in one or more of the floating-point calculations performed by the instruction, regardless of the value of the IXE bit.

#### **UFC, bit [3]**

Underflow cumulative floating-point exception bit. This bit is set to 1 to indicate that the Underflow floating-point exception has occurred since 0 was last written to this bit.

How VFP instructions update this bit depends on the value of the UFE bit.

Advanced SIMD instructions set this bit if the Underflow floating-point exception occurs in one or more of the floating-point calculations performed by the instruction, regardless of the value of the UFE bit.

#### **OFC, bit [2]**

Overflow cumulative floating-point exception bit. This bit is set to 1 to indicate that the Overflow floating-point exception has occurred since 0 was last written to this bit.

How VFP instructions update this bit depends on the value of the OFE bit.

Advanced SIMD instructions set this bit if the Overflow floating-point exception occurs in one or more of the floating-point calculations performed by the instruction, regardless of the value of the OFE bit.

#### **DZC, bit [1]**

Divide by Zero cumulative floating-point exception bit. This bit is set to 1 to indicate that the Divide by Zero floating-point exception has occurred since 0 was last written to this bit.

How VFP instructions update this bit depends on the value of the DZE bit.

Advanced SIMD instructions set this bit if the Divide by Zero floating-point exception occurs in one or more of the floating-point calculations performed by the instruction, regardless of the value of the DZE bit.

#### **IOC, bit [0]**

Invalid Operation cumulative floating-point exception bit. This bit is set to 1 to indicate that the Invalid Operation floating-point exception has occurred since 0 was last written to this bit.

How VFP instructions update this bit depends on the value of the IOE bit.

Advanced SIMD instructions set this bit if the Invalid Operation floating-point exception occurs in one or more of the floating-point calculations performed by the instruction, regardless of the value of the IOE bit.

## Accessing the FPSCR

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

```
VMRS <Rt>, <spec_reg>
```

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

```
VMSR <spec_reg>, <Rt>
```

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

<spec_reg>	reg
FPSCR	0001

## 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<sup>®</sup> 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 [CPACR.cp10](#)==00, accesses to this register from PL0 and PL1 are UNDEFINED.
- If [CPACR.cp10](#)==01, accesses to this register from PL0 are UNDEFINED.
- If [CPACR\\_EL1.FPEN](#)==00, accesses to this register from PL0 are trapped to EL1.
- If [CPACR\\_EL1.FPEN](#)==01, accesses to this register from PL0 are trapped to EL1.
- If [CPACR\\_EL1.FPEN](#)==10, accesses to this register from PL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [CPTR\\_EL2.TFP](#)==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 [CPACR\\_EL1.FPEN](#)==00, Non-secure accesses to this register from EL0 are trapped to EL1.
- If [CPACR\\_EL1.FPEN](#)==01, Non-secure accesses to this register from EL0 are trapped to EL1.
- If [CPACR\\_EL1.FPEN](#)==10, Non-secure accesses to this register from EL0 are trapped to EL1.
- If [CPTR\\_EL2.FPEN](#)==00, Non-secure accesses to this register from EL0 and EL1 are trapped to EL2.

- If [CPTR\\_EL2](#).FPEN==10, 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==1 :

- If [CPTR\\_EL2](#).FPEN==00, Non-secure accesses to this register from EL0 are trapped to EL2.
- If [CPTR\\_EL2](#).FPEN==01, Non-secure accesses to this register from EL0 are trapped to EL2.
- If [CPTR\\_EL2](#).FPEN==10, 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 [HCPTR](#).TCP10==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.
- If [HCPTR](#).TCP10==1, Non-secure accesses to this register from EL2 are UNDEFINED.

When EL3 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [NSACR](#).cp10==0, Non-secure accesses to this register from EL0, EL1, and EL2 are UNDEFINED.

When EL3 is implemented and is using AArch64 :

- If [CPTR\\_EL3](#).TFP==1, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

02/05/2017 15:43

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

# FPSID, Floating-Point System ID register

The FPSID characteristics are:

## Purpose

Provides top-level information about the floating-point implementation.

This register largely duplicates information held in the [MIDR](#). ARM deprecates use of it.

This register is part of:

- The Floating-point registers functional group.
- The Identification registers functional group.

## Configuration

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

Implemented only if the implementation includes the Advanced SIMD and floating-point functionality.

## Attributes

FPSID is a 32-bit register.

## Field descriptions

The FPSID 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								SW	Subarchitecture							PartNum						Variant			Revision						

### Implementer, bits [31:24]

Implementer codes are the same as those used for the [MIDR](#).

For an implementation by ARM this field is 0x41, the ASCII code for A.

### SW, bit [23]

Software bit. Defined values are:

SW	Meaning
0	The implementation provides a hardware implementation of the floating-point instructions.
1	The implementation supports only software emulation of the floating-point instructions.

In ARMv8-A the only permitted value is 0.

### Subarchitecture, bits [22:16]

Subarchitecture version number. For an implementation by ARM, defined values are:

Subarchitecture	Meaning
0000000	VFPv1 architecture with an IMPLEMENTATION DEFINED subarchitecture.
0000001	VFPv2 architecture with Common VFP subarchitecture v1.
0000010	VFPv3 architecture, or later, with Common VFP subarchitecture v2. The VFP architecture version is indicated by the <a href="#">MVFR0</a> and <a href="#">MVFR1</a> registers.
0000011	VFPv3 architecture, or later, with Null subarchitecture. The entire floating-point implementation is in hardware, and no software support code is required. The VFP architecture version is indicated by the <a href="#">MVFR0</a> and <a href="#">MVFR1</a> registers. This value can be used only by an implementation that does not support the trap enable bits in the <a href="#">FPSCR</a> .
0000100	VFPv3 architecture, or later, with Common VFP subarchitecture v3, and support for trap enable bits in <a href="#">FPSCR</a> . The VFP architecture version is indicated by the <a href="#">MVFR0</a> and <a href="#">MVFR1</a> registers.

For a subarchitecture designed by ARM the most significant bit of this field, register bit[22], is 0. Values with a most significant bit of 0 that are not listed here are reserved.

When the subarchitecture designer is not ARM, the most significant bit of this field, register bit[22], must be 1. Each implementer must maintain its own list of subarchitectures it has designed, starting at subarchitecture version number 0×40.

In ARMv8-A the permitted values are 0000011 and 0000100.

### PartNum, bits [15:8]

An IMPLEMENTATION DEFINED part number for the floating-point implementation, assigned by the implementer.

### Variant, bits [7:4]

An IMPLEMENTATION DEFINED variant number. Typically, this field distinguishes between different production variants of a single product.

### Revision, bits [3:0]

An IMPLEMENTATION DEFINED revision number for the floating-point implementation.

## Accessing the FPSID

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

```
VMRS <Rt>, <spec_reg>
```

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

```
VMSR <spec_reg>, <Rt>
```

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

<spec_reg>	reg
FPSID	0000

## 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.

When access to this register is permitted, write accesses are ignored.

## 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 `CPACR.cp10==00`, accesses to this register from PL1 are UNDEFINED.

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

- If `CPTR_EL2.TFP==1`, Non-secure accesses to this register from EL1 are trapped to EL2.
- If `HCR_EL2.TID0==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 `CPTR_EL2.FPEN==00`, Non-secure accesses to this register from EL1 are trapped to EL2.
- If `CPTR_EL2.FPEN==10`, Non-secure accesses to this register from EL1 are trapped to EL2.
- If `HCR_EL2.TID0==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 `HCPTR.TCP10==1`, Non-secure accesses to this register from EL1 are trapped to Hyp mode.
- If `HCPTR.TCP10==1`, Non-secure accesses to this register from EL2 are UNDEFINED.
- If `HCR.TID0==1`, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

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

- If `NSACR.cp10==0`, Non-secure accesses to this register from EL1 and EL2 are UNDEFINED.

When EL3 is implemented and is using AArch64 :

- If `CPTR_EL3.TFP==1`, accesses to this register from EL1 and EL2 are trapped to EL3.



# HACR, Hyp Auxiliary Configuration Register

The HACR characteristics are:

## Purpose

Controls trapping to Hyp mode of IMPLEMENTATION DEFINED aspects of Non-secure EL1 or EL0 operation.

This register is part of the Virtualization registers functional group.

## Configuration

AArch32 System register HACR is architecturally mapped to AArch64 System register [HACR\\_EL2](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HACR is a 32-bit register.

## Field descriptions

The HACR 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the HACR

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>, c1, c1, 7	100	111	0001	1111	0001

## 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<sup>®</sup> 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.

02/05/2017 15:43

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

# HACTLR, Hyp Auxiliary Control Register

The HACTLR characteristics are:

## Purpose

Controls IMPLEMENTATION DEFINED features of Hyp mode operation.

This register is part of:

- The Virtualization registers functional group.
- The Other system control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch32 System register HACTLR is architecturally mapped to AArch64 System register [ACTLR\\_EL2\[31:0\]](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HACTLR is a 32-bit register.

## Field descriptions

The HACTLR 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the HACTLR

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>, c1, c0, 1	100	001	0001	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.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.

02/05/2017 15:43

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

# HACTLR2, Hyp Auxiliary Control Register 2

The HACTLR2 characteristics are:

## Purpose

Provides additional space to the HACTLR register to hold IMPLEMENTATION DEFINED trap functionality.

This register is part of:

- The Virtualization registers functional group.
- The Other system control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch32 System register HACTLR2 is architecturally mapped to AArch64 System register [ACTLR\\_EL2\[63:32\]](#).

In ARMv8.0 and ARMv8.1, it is IMPLEMENTATION DEFINED whether this register is implemented, or whether it causes UNDEFINED exceptions when accessed. The implementation of this register can be detected by examining [ID\\_MMFR4.AC2](#).

From ARMv8.2 this register must be implemented.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HACTLR2 is a 32-bit register.

## Field descriptions

The HACTLR2 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the HACTLR2

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>, c1, c0, 3	100	011	0001	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<sup>®</sup> 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.

02/05/2017 15:43

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

# HADFSR, Hyp Auxiliary Data Fault Status Register

The HADFSR characteristics are:

## Purpose

Provides additional IMPLEMENTATION DEFINED syndrome information for Data Abort exceptions taken to Hyp mode.

This register is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch32 System register HADFSR is architecturally mapped to AArch64 System register [AFSR0\\_EL2](#).

This is an optional register. An implementation that does not require this register can implement it as RES0.

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HADFSR is a 32-bit register.

## Field descriptions

The HADFSR 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the HADFSR

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>, c5, c1, 0	100	000	0101	1111	0001

## 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<sup>®</sup> 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](#).T5==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](#).T5==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](#).T5==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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



# HAIFSR, Hyp Auxiliary Instruction Fault Status Register

The HAIFSR characteristics are:

## Purpose

Provides additional IMPLEMENTATION DEFINED syndrome information for Prefetch Abort exceptions taken to Hyp mode.

This register is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch32 System register HAIFSR is architecturally mapped to AArch64 System register [AFSR1\\_EL2](#).

This is an optional register. An implementation that does not require this register can implement it as RES0.

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HAIFSR is a 32-bit register.

## Field descriptions

The HAIFSR 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the HAIFSR

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>, c5, c1, 1	100	001	0101	1111	0001

## 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<sup>®</sup> 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](#).T5==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](#).T5==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](#).T5==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# HMAIR0, Hyp Auxiliary Memory Attribute Indirection Register 0

The HMAIR0 characteristics are:

## Purpose

Provides IMPLEMENTATION DEFINED memory attributes for the memory attribute encodings defined by [HMAIR0](#). These IMPLEMENTATION DEFINED attributes can only provide additional qualifiers for the memory attribute encodings, and cannot change the memory attributes defined in [HMAIR0](#).

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch32 System register HMAIR0 is architecturally mapped to AArch64 System register [AMAIR\\_EL2\[31:0\]](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HMAIR0 is a 32-bit register.

## Field descriptions

The HMAIR0 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
IMPLEMENTATION DEFINED																															

If an implementation does not provide any IMPLEMENTATION DEFINED memory attributes, this register is RES0.

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the HMAIR0

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>, c10, c3, 0	100	000	1010	1111	0011

## 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<sup>®</sup> 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.T10](#)==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.T10](#)==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.T10](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# HMAIR1, Hyp Auxiliary Memory Attribute Indirection Register 1

The HMAIR1 characteristics are:

## Purpose

Provides IMPLEMENTATION DEFINED memory attributes for the memory attribute encodings defined by [HMAIR1](#). These IMPLEMENTATION DEFINED attributes can only provide additional qualifiers for the memory attribute encodings, and cannot change the memory attributes defined in [HMAIR1](#).

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch32 System register HMAIR1 is architecturally mapped to AArch64 System register [AMAIR\\_EL2\[63:32\]](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HMAIR1 is a 32-bit register.

## Field descriptions

The HMAIR1 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
IMPLEMENTATION DEFINED																															

If an implementation does not provide any IMPLEMENTATION DEFINED memory attributes, this register is RES0.

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the HMAIR1

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>, c10, c3, 1	100	001	1010	1111	0011

## 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<sup>®</sup> 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.T10==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.T10==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.T10==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# HCPTR, Hyp Architectural Feature Trap Register

The HCPTR characteristics are:

## Purpose

Controls:

- Trapping to Hyp mode of Non-secure access, at EL1 or EL0, to trace, and to Advanced SIMD and floating-point functionality.
- Hyp mode access to trace, and to Advanced SIMD and floating-point functionality.

### Note

Accesses to this functionality:

- From Non-secure modes other than Hyp mode are also affected by settings in the [CPACR](#) and [NSACR](#).
- From Hyp mode are also affected by settings in the [NSACR](#).

Exceptions generated by the [CPACR](#) and [NSACR](#) controls are higher priority than those generated by the HCPTR controls.

This register is part of the Virtualization registers functional group.

## Configuration

AArch32 System register HCPTR is architecturally mapped to AArch64 System register [CPTR\\_EL2](#).

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 with EL2 using AArch32, or into EL3 with EL3 using AArch32. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HCPTR is a 32-bit register.

## Field descriptions

The HCPTR 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
<a href="#">TCPAC</a>	0	0	0	0	0	0	0	0	0	0	0	<a href="#">TTA</a>	0	0	0	0	<a href="#">TASE</a>	0	1	1	<a href="#">TCP11</a>	<a href="#">TCP10</a>	1	1	1	1	1	1	1	1	1	1

### TCPAC, bit [31]

Traps Non-secure EL1 accesses to the [CPACR](#) to Hyp mode.

TCPAC	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 accesses to the <a href="#">CPACR</a> are trapped to Hyp mode.

### Note

The [CPACR](#) is not accessible at EL0.

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

**Bits [30:21]**

Reserved, RES0.

**TTA, bit [20]**

Traps Non-secure System register accesses to all implemented trace registers to Hyp mode.

TTA	Meaning
0	This control does not cause any instructions to be trapped.
1	Any Non-secure System register access to an implemented trace register is trapped to Hyp mode, unless the access is trapped to EL1 by a <a href="#">CPACR</a> or <a href="#">NSACR</a> control, or the access is from Non-secure EL0 and the definition of the register in the appropriate trace architecture specification indicates that the register is not accessible from EL0. A trapped instruction generates: <ul style="list-style-type: none"> <li>• A Hyp Trap exception, if the exception is taken from Non-secure EL0 or EL1.</li> <li>• An Undefined Instruction exception taken to Hyp mode, if the exception is taken from Hyp mode.</li> </ul>

If the implementation does not include a trace macrocell, or does not include a System register interface to the trace macrocell registers, it is IMPLEMENTATION DEFINED whether this bit:

- Is RES0.
- Is RES1.
- Can be written from Hyp mode, and from Secure Monitor mode when [SCR.NS](#) is 1.

If EL3 is implemented and is using AArch32, and the value of [NSACR.NSTRCDIS](#) is 1, in Non-secure state this field behaves as RAO/WI, regardless of its actual value.

**Note**

- The ETMv4 architecture does not permit EL0 to access the trace registers. If the implementation includes an ETMv4 implementation, EL0 accesses to the trace registers are UNDEFINED, and a resulting Undefined Instruction exception is higher priority than a HCPTR.TTA Hyp Trap exception.
- The architecture does not provide traps on trace register accesses through the optional memory-mapped external debug interface.

System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**Bits [19:16]**

Reserved, RES0.

**TASE, bit [15]**

Traps Non-secure execution of Advanced SIMD instructions to Hyp mode when the value of HCPTR.TCP10 is 0.

TASE	Meaning
0	This control does not cause any instructions to be trapped.
1	When the value of HCPTR.TCP10 is 0, any attempt to execute an Advanced SIMD instruction in Non-secure state is trapped to Hyp mode, unless it is trapped to EL1 by a <a href="#">CPACR</a> or <a href="#">NSACR</a> control. A trapped instruction generates: <ul style="list-style-type: none"> <li>• A Hyp Trap exception, if the exception is taken from Non-secure EL0 or EL1.</li> <li>• An Undefined Instruction exception taken to Hyp mode, if the exception is taken from Hyp mode.</li> </ul>

When the value of HCPTR.TCP10 is 1, the value of this field is ignored.

If the implementation does not include Advanced SIMD and floating-point functionality, this field is RES1. Otherwise, it is IMPLEMENTATION DEFINED whether this field is implemented as a RW field. If it is not implemented as a RW field, then it is RAZ/WI.



If EL3 is implemented and is using AArch32, and the value of [NSACR](#).NSASEDIS is 1, in Non-secure state this field behaves as RAO/WI, regardless of its actual value. This applies even if the field is implemented as RAZ/WI.

For the list of instructions affected by this field, see 'Controls of Advanced SIMD operation that do not apply to floating-point operation' in the ARMv8 ARM, section E1.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### Bit [14]

Reserved, RES0.

#### Bits [13:12]

Reserved, RES1.

#### TCP11, bit [11]

The value of this field is ignored. If this field is programmed with a different value to the TCP10 bit then this field is UNKNOWN on a direct read of the HCPTR.

If the implementation does not include Advanced SIMD and floating-point functionality, this field is RES1.

If EL3 is implemented and is using AArch32, and the value of [NSACR](#).cp10 is 0, in Non-secure state this field behaves as RAO/WI, regardless of its actual value.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### TCP10, bit [10]

Trap Non-secure accesses to Advanced SIMD and floating-point functionality to Hyp mode:

TCP10	Meaning
0	This control does not cause any instructions to be trapped.
1	Any attempted access to Advanced SIMD and floating-point functionality from Non-secure state is trapped to Hyp mode, unless it is trapped to EL1 by a <a href="#">CPACR</a> or <a href="#">NSACR</a> control. A trapped instruction generates: <ul style="list-style-type: none"> <li>A Hyp Trap exception, if the exception is taken from Non-secure EL0 or EL1.</li> <li>An Undefined Instruction exception taken to Hyp mode, if the exception is taken from Hyp mode.</li> </ul>

The Advanced SIMD and floating-point features controlled by these fields are:

- Execution of any floating-point or Advanced SIMD instruction.
- Any access to the Advanced SIMD and floating-point registers D0-D31 and their views as S0-S31 and Q0-Q15.
- Any access to the [FPSCR](#), [FPSID](#), [MVFR0](#), [MVFR1](#), [MVFR2](#), or [FPEXC](#) System registers.

If the implementation does not include Advanced SIMD and floating-point functionality, this field is RES1.

If EL3 is implemented and is using AArch32, and the value of [NSACR](#).cp10 is 0, in Non-secure state this field behaves as RAO/WI, regardless of its actual value.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### Bits [9:0]

Reserved, RES1.

## Accessing the HCPTR

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>, c1, c1, 2	100	010	0001	1111	0001

## 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<sup>®</sup> 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 EL3 is implemented and is using AArch64 :

- If [CPTR\\_EL3.TCPAC](#)==1, accesses to this register from EL2 are trapped to EL3.

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.

02/05/2017 15:43

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

# HCR, Hyp Configuration Register

The HCR characteristics are:

## Purpose

Provides configuration controls for virtualization, including defining whether various Non-secure operations are trapped to Hyp mode.

This register is part of the Virtualization registers functional group.

## Configuration

AArch32 System register HCR is architecturally mapped to AArch64 System register [HCR\\_EL2\[31:0\]](#).

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 with EL2 using AArch32, or into EL3 with EL3 using AArch32. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HCR is a 32-bit register.

## Field descriptions

The HCR 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
0	TRVM	HCD	0	TGETVM	TTLB	TPU	TPC	TSW	TACT	IDCPT	TSC	TID3	TID2	TID1	TID0	TWE	TWI	DCBSU	FBVA	VIVF	AMO	IMO	FMO	PTV					

### Bit [31]

Reserved, RES0.

### TRVM, bit [30]

Trap Reads of Virtual Memory controls. Traps Non-secure EL1 reads of the virtual memory control registers to Hyp mode. The registers for which read accesses are trapped are as follows:

[SCTLR](#), [TTBR0](#), [TTBR1](#), [TTBCR](#), [TTBCR2](#), [DACR](#), [DFSR](#), [IFSR](#), [DFAR](#), [IFAR](#), [ADFSR](#), [AIFSR](#), [PRRR](#), [NMRR](#), [MAIR0](#), [MAIR1](#), [AMAIRO](#), [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 Hyp mode.

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

### HCD, bit [29]

HVC instruction disable. Disables Non-secure state execution of HVC instructions.

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. The Undefined Instruction exception is taken to the Exception level at which the HVC instruction is executed.

**Note**

HVC instructions are always UNDEFINED at EL0.

This bit is only implemented if EL3 is not implemented. Otherwise, it is RES0.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**Bit [28]**

Reserved, RES0.

**TGE, bit [27]**

Trap General Exceptions, from Non-secure EL0.

TGE	Meaning
0	This control has no effect on execution at EL0.
1	When the value of <a href="#">SCR.NS</a> is 0, this control has no effect on execution at EL0. When the value of <a href="#">SCR.NS</a> is 1, then: <ul style="list-style-type: none"> <li>All exceptions that would be routed to EL1 are routed to EL2.</li> <li>The <a href="#">SCTLR.M</a> bit is treated as being 0 for all purposes other than returning the result of a direct read of <a href="#">SCTLR</a>.</li> <li>The HCR.{FMO, IMO, AMO} bits are treated as being 1 for all purposes other than returning the result of a direct read of HCR.</li> <li>All virtual interrupts are disabled.</li> <li>Any IMPLEMENTATION DEFINED mechanisms for signaling virtual interrupts are disabled.</li> <li>An exception return to EL1 is treated as an illegal exception return.</li> <li>Monitor mode execution of an MSR or CPS instruction that changes <a href="#">CPSR.M</a> to a Non-secure EL1 mode is an illegal change to PSTATE.M. For more information see 'Illegal changes to PSTATE.M' in the ARMv8 ARM, section G1 (The AArch32 System Level Programmers' Model).</li> </ul>

Also, when HCR.TGE is 1:

- If EL3 is using AArch32, an attempt to change from a Secure PL1 mode to a Non-secure EL1 mode by changing [SCR.NS](#) from 0 to 1 results in [SCR.NS](#) remaining as 0.
- The [HDCR](#).{TDRA, TDOSA, TDA, TDE} bits are ignored and treated as being 1 other than for the purpose of a direct read of [HDCR](#).

In the following cases the field resets to 0:

- The PE resets into EL3 with EL3 using AArch32.
- The PE resets into EL2 with EL2 using AArch32.

Otherwise, the field reset value is architecturally UNKNOWN.

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

**TVM, bit [26]**

Trap Virtual Memory controls. Traps Non-secure EL1 writes to the virtual memory control registers to Hyp mode. The registers for which write accesses are trapped are as follows:

[SCTLR](#), [TTBR0](#), [TTBR1](#), [TTBCR](#), [TTBCR2](#), [DACR](#), [DFSR](#), [IFSR](#), [DFAR](#), [IFAR](#), [ADFSR](#), [AIFSR](#), [PRRR](#), [NMRR](#), [MAIR0](#), [MAIR1](#), [AMAIRO](#), [AMAIR1](#), [CONTEXTIDR](#).

TVM	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 write accesses to EL1 virtual memory control registers are trapped to Hyp mode.

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

**TTLB, bit [25]**

Trap TLB maintenance instructions. Traps Non-secure EL1 execution of a TLBI instruction to Hyp mode. This applies to the following instructions:

[TLBIALLIS](#), [TLBIMVAIS](#), [TLBIASIDIS](#), [TLBIMVAAIS](#), [TLBIMVALIS](#), [TLBIMVAALIS](#), [ITLBIALL](#), [ITLBIMVA](#), [ITLBIASID](#), [DTLBIALL](#), [DTLBIMVA](#), [DTLBIASID](#), [TLBIALL](#), [TLBIMVA](#), [TLBIASID](#), [TLBIMVAA](#), [TLBIMVAL](#), [TLBIMVAAL](#)

TTLB	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 accesses to TLB maintenance instructions are trapped to Hyp mode.

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

**TPU, bit [24]**

Trap cache maintenance instructions that operate to the Point of Unification. Traps Non-secure EL1 execution of those cache maintenance instructions to Hyp mode. This applies to the following instructions:

[ICIMVAU](#), [ICIALLU](#), [ICIALLUIS](#), [DCCMVAU](#).

**Note**

An Undefined Instruction exception generated at EL0 is higher priority than this trap to EL2, and these instructions are always UNDEFINED at EL0.

TPU	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 execution of the specified instructions is trapped to Hyp mode.

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

**TPC, bit [23]**

Trap data or unified cache maintenance instructions that operate to the Point of Coherency. Traps Non-secure EL1 execution of those cache maintenance instructions to Hyp mode. This applies to the following instructions:

[DCIMVAC](#), [DCCIMVAC](#), [DCCMVAC](#).

**Note**

An Undefined Instruction exception generated at EL0 is higher priority than this trap to EL2, and these instructions are always UNDEFINED at EL0.

TPC	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure execution of the specified instructions is trapped to Hyp mode.

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

**TSW, bit [22]**

Trap data or unified cache maintenance instructions that operate by Set/Way. Traps Non-secure EL1 execution of those cache maintenance instructions by set/way to Hyp mode. This applies to the following instructions:

[DCISW](#), [DCCSW](#), [DCCISW](#).

**Note**

An Undefined Instruction exception generated at EL0 is higher priority than this trap to EL2, and these instructions are always UNDEFINED at EL0.

TSW	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure execution of the specified instructions is trapped to Hyp mode.

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

### TAC, bit [21]

Trap Auxiliary Control Registers. Traps Non-secure EL1 accesses to the Auxiliary Control Registers to Hyp mode, from both Execution states. This applies to the following register accesses:

[ACTLR](#) and, if implemented, [ACTLR2](#).

TAC	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 accesses to the specified registers are trapped to Hyp mode.

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

### TIDCP, bit [20]

Trap IMPLEMENTATION DEFINED functionality. Traps Non-secure EL1 accesses to the encodings for IMPLEMENTATION DEFINED System Registers to Hyp mode.

MCR and MRC instructions accessing the following encodings:

- All coproc==p15, CRn==c9, Opcode1 = {0-7}, CRm == {c0-c2, c5-c8}, opcode2 == {0-7}.
- All coproc==p15, CRn==c10, Opcode1 == {0-7}, CRm == {c0, c1, c4, c8}, opcode2 == {0-7}.
- All coproc==p15, CRn==c11, Opcode1 == {0-7}, CRm == {c0-c8, c15}, opcode2 == {0-7}.

When HCR.TIDCP is set to 1, it is IMPLEMENTATION DEFINED whether any of this functionality accessed from Non-secure EL0 is trapped to Hyp mode. If it is not, it is UNDEFINED, and the PE takes an Undefined Instruction exception to Non-secure Undefined mode.

TIDCP	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 accesses to the specified System register encodings for IMPLEMENTATION DEFINED functionality are trapped to Hyp mode.

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

### TSC, bit [19]

Trap SMC instructions. Traps Non-secure EL1 execution of SMC instructions to Hyp mode.

TSC	Meaning
0	This control does not cause any instructions to be trapped.
1	Any attempt to execute an SMC instruction at Non-secure EL1 is trapped to Hyp mode, regardless of the value of <a href="#">SCR.SCD</a> .

The ARMv8-A architecture permits, but does not require, this trap to apply to conditional SMC instructions that fail their condition code check, in the same way as with traps on other conditional instructions.

#### Note

- This trap is only implemented if the implementation includes EL3.
- SMC instructions are always UNDEFINED at PL0.
- This bit traps execution of the SMC instruction. It is not a routing control for the SMC exception. Hyp Trap exceptions and SMC exceptions have different preferred return addresses.

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

### TID3, bit [18]

Trap ID group 3. Traps Non-secure EL1 reads of the following registers to Hyp mode:

[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.

Also an MRC access to any of the following encodings:

- 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 Hyp mode.

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

## TID2, bit [17]

Trap ID group 2. Traps the following register accesses to Hyp mode:

- Non-secure EL1 and EL0 reads of the [CTR](#), [CCSIDR](#), [CLIDR](#), and [CSSELR](#).
- Non-secure EL1 and EL0 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 Hyp mode.

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

## TID1, bit [16]

Trap ID group 1. Traps Non-secure EL1 reads of the following registers to Hyp mode:

[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 Hyp mode.

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

## TID0, bit [15]

Trap ID group 0. Traps the following register accesses to Hyp mode:

- Non-secure EL1 reads of the [JIDR](#) and [FPSID](#).
- If the [JIDR](#) is RAZ from Non-secure EL0, Non-secure EL0 reads of the [JIDR](#).

### Note

- It is IMPLEMENTATION DEFINED whether the [JIDR](#) is RAZ or UNDEFINED at EL0. If it is UNDEFINED at EL0 then the Undefined Instruction exception takes precedence over this trap.
- The [FPSID](#) is not accessible at EL0.
- Writes to the [FPSID](#) are ignored, and not trapped by this control.

TID0	Meaning
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 Hyp mode.

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

#### TWE, bit [14]

Traps Non-secure EL0 and EL1 execution of WFE instructions to Hyp mode:

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 Hyp mode, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by <a href="#">SCTLR.nTWE</a> .

The attempted execution of a conditional WFE instruction is only trapped if the instruction passes its condition code check.

#### Note

Since a WFE can complete at any time, even without a Wakeup event, the traps on WFE are not guaranteed to be taken, even if the WFE is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

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

#### TWI, bit [13]

Traps Non-secure EL0 and EL1 execution of WFI instructions to Hyp mode.

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 Hyp mode, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by <a href="#">SCTLR.nTWI</a> .

The attempted execution of a conditional WFI instruction is only trapped if the instruction passes its condition code check.

#### Note

Since a WFI can complete at any time, even without a Wakeup event, the traps on WFI are not guaranteed to be taken, even if the WFI is executed when there is no Wakeup event. The only guarantee is that if the instruction does not complete in finite time in the absence of a Wakeup event, the trap will be taken.

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

#### 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"> <li>The <a href="#">SCTLR.M</a> field behaves as 0 for all purposes other than a direct read of the value of the field.</li> <li>The HCR.VM field behaves as 1 for all purposes other than a direct read of the value of the field.</li> <li>The memory type produced by the first stage of the EL1&amp;0 translation regime is Normal Non-Shareable, Inner Write-Back Read-Allocate Write-Allocate, Outer Write-Back Read-Allocate Write-Allocate.</li> </ul>



This field has no effect on the EL2 and EL3 translation regimes.

This field is permitted to be cached in a TLB.

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

### BSU, bits [11:10]

Barrier Shareability upgrade. This field determines the minimum shareability domain that is applied to any barrier instruction executed from Non-secure EL1 or Non-secure EL0:

BSU	Meaning
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.

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

### FB, bit [9]

Force broadcast. Causes the following instructions to be broadcast within the Inner Shareable domain when executed from Non-secure EL1:

[BPIALL](#), [TLBIALL](#), [TLBIMVA](#), [TLBIASID](#), [DTLBIALL](#), [DTLBIMVA](#), [DTLBIASID](#), [ITLBIALL](#), [ITLBIMVA](#), [ITLBIASID](#), [TLBIMVAA](#), [ICIALLU](#), [TLBIMVAL](#), [TLBIMVAAL](#).

FB	Meaning
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.

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

### VA, bit [8]

Virtual SError interrupt exception.

VA	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 enabled only when the value of HCR.{TGE, AMO} is {0, 1}.

The Guest OS cannot distinguish the virtual exception from the corresponding physical exception.

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

### VI, bit [7]

Virtual IRQ exception.

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.{TGE, IMO} is {0, 1}.

The Guest OS cannot distinguish the virtual exception from the corresponding physical exception.

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

**VF, bit [6]**

Virtual FIQ exception.

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.{TGE, FMO} is {0, 1}.

The Guest OS cannot distinguish the virtual exception from the corresponding physical exception.

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

**AMO, bit [5]**

Error interrupt Mask Override. When this bit is set to 1, it overrides the effect of [CPSR.A](#), and enables virtual exception signaling by the VA bit.

If the value of HCR.TGE is 0, then Virtual Error Interrupts are enabled in Non-secure state.

If the value of HCR.TGE is 1, then in Non-secure state the HCR.AMO bit behaves as 1 for all purposes other than a direct read of the value of the bit.

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

**IMO, bit [4]**

IRQ Mask Override. When this bit is set to 1, it overrides the effect of [CPSR.I](#), and enables virtual exception signaling by the VI bit.

If the value of HCR.TGE is 0, then Virtual IRQ interrupts are enabled in the Non-secure state.

If the value of HCR.TGE is 1, then in Non-secure state the HCR.IMO bit behaves as 1 for all purposes other than a direct read of the value of the bit.

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

**FMO, bit [3]**

FIQ Mask Override. When this bit is set to 1, it overrides the effect of [CPSR.F](#), and enables virtual exception signaling by the VF bit.

If the value of HCR.TGE is 0, then Virtual FIQ interrupts are enabled in the Non-secure state.

If the value of HCR.TGE is 1, then in Non-secure state the HCR.FMO bit behaves as 1 for all purposes other than a direct read of the value of the bit.

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

**PTW, bit [2]**

Protected Table Walk. In the Non-secure PL1&0 translation regime, a translation table access made as part of a stage 1 translation table walk is subject to a stage 2 translation. The combining of the memory type attributes from the two stages of translation means the access might be made to a type of Device memory. If this occurs then the value of this bit determines the behavior:

PTW	Meaning
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.

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

**SWIO, bit [1]**

Set/Way Invalidation Override. Causes Non-secure EL1 execution of the data cache invalidate by set/way instructions to perform a data cache clean and invalidate by set/way.

SWIO	Meaning
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 this bit is set to 1, [DCISW](#) performs the same invalidation as a [DCCISW](#) instruction.

As a result of changes to the behavior of [DCISW](#), this bit is redundant in ARMv8. This bit can be implemented as RES1.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**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.

If the HCR.DC bit is set to 1, then the behavior of the PE when executing in a Non-secure mode other than Hyp mode is consistent with HCR.VM being 1, regardless of the actual value of HCR.VM, other than the value returned by an explicit read of HCR.VM.

When the value of this bit is 1, data cache invalidate instructions executed at Non-secure EL1 perform a data cache clean and invalidate. For the invalidate by set/way instruction this behavior applies regardless of the value of the HCR.SWIO bit.

This bit is permitted to be cached in a TLB.

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

**Accessing the HCR**

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>, c1, c1, 0	100	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	-
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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T1](#)=1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# HCR2, Hyp Configuration Register 2

The HCR2 characteristics are:

## Purpose

Provides additional configuration controls for virtualization.

This register is part of the Virtualization registers functional group.

## Configuration

AArch32 System register HCR2 is architecturally mapped to AArch64 System register [HCR\\_EL2\[63:32\]](#).

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 with EL2 using AArch32, or into EL3 with EL3 using AArch32. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HCR2 is a 32-bit register.

## Field descriptions

The HCR2 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	MIO	CN	CE	TE	A	TERR	0	0	ID	CD

### Bits [31:7]

Reserved, RES0.

### MIOCNE, bit [6]

Mismatched Inner/Outer Cacheable Non-Coherency Enable, for the Non-secure PL1&0 translation regime.

MIOCNE	Meaning
0	For the Non-secure PL1&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 PL1&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 E2 (The AArch32 Application Level Memory Model).

The value of this field has no effect on translation regimes other than the Non-secure PL1&0 translation regime.

This field can be implemented as 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.

**TEA, bit [5]**

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.

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

**TERR, bit [4]**

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.

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

**Bits [3:2]**

Reserved, RES0.

**ID, bit [1]**

Stage 2 Instruction access cacheability disable. For the Non-secure PL1&0 translation regime, when [HCR.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 PL1&0 translation regime.
1	For the Non-secure PL1&0 translation regime, forces all stage 2 translations for instruction accesses to Normal memory to be Non-cacheable.

This bit has no effect on the EL2 translation regime.

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

**CD, bit [0]**

Stage 2 Data access cacheability disable. When [HCR.VM](#)==1, this forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable for the Non-secure PL1&0 translation regime.

CD	Meaning
0	This control has no effect on stage 2 of the Non-secure PL1&0 translation regime for data accesses and translation table walks.
1	For the Non-secure PL1&0 translation regime, forces all stage 2 translations for data accesses and translation table walks to Normal memory to be Non-cacheable.

This bit has no effect on the EL2 translation regime.

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

## Accessing the HCR2

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>, c1, c1, 4	100	100	0001	1111	0001

## 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<sup>®</sup> 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.

# HDCR, Hyp Debug Control Register

The HDCR characteristics are:

## Purpose

Controls the trapping to Hyp mode of Non-secure accesses, at EL1 or lower, to functions provided by the debug and trace architectures and the Performance Monitors Extension.

This register is part of:

- The Debug registers functional group.
- The Virtualization registers functional group.

## Configuration

AArch32 System register HDCR is architecturally mapped to AArch64 System register [MDCR\\_EL2](#).

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

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

HDCR is a 32-bit register.

## Field descriptions

The HDCR 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	HPMD	0	0	0	0	0	TDRA	TDOSA	TDATDE	HPME	TPM	TPMCR					HPMN

### Bits [31:18]

Reserved, RES0.

### HPMD, bit [17]

In ARMv8.2 and ARMv8.1:

Guest Performance Monitors Disable. This control prohibits event counting at EL2. Permitted values are:

HPMD	Meaning
0	Event counting allowed in Hyp mode.
1	Event counting prohibited in Hyp mode. In an ARMv8.1 implementation, event counting is prohibited unless enabled by the IMPLEMENTATION DEFINED authentication interface ExternalSecureNoninvasiveDebugEnabled().

This control applies only to:

- The event counters in the range [0..(HPMN-1)].
- If [PMCR.DP](#) is set to 1, [PMCCNTR](#).

The other event counters are unaffected. When [PMCR.DP](#) is set to 0, [PMCCNTR](#) is unaffected.

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



**In ARMv8.0:**

Reserved, RES0.

**Bits [16:12]**

Reserved, RES0.

**TDRA, bit [11]**

Trap Debug ROM Address register access. Traps Non-secure EL0 and EL1 System register accesses to the Debug ROM registers to Hyp mode.

TDRA	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL0 and EL1 System register accesses to the <a href="#">DBGDRAR</a> or <a href="#">DBGDSAR</a> are trapped to Hyp mode, unless it is trapped by <a href="#">DBGDSCRext.UDCCdis</a> .

If [HCR.TGE](#) or [HDCR.TDE](#) is 1, behavior is as if this bit is 1 other than for the purpose of a direct read.

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

**TDOSA, bit [10]**

Trap debug OS-related register access. Traps Non-secure EL1 System register accesses to the powerdown debug registers to Hyp mode.

TDOSA	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 System register accesses to the powerdown debug registers are trapped to Hyp mode.

The registers for which accesses are trapped are as follows:

- [DBGOSLSR](#), [DBGOSLAR](#), [DBGOSDLR](#), and the [DBGPRCR](#).
- Any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit.

**Note**

These registers are not accessible at EL0.

If [HCR.TGE](#) or [HDCR.TDE](#) is 1, behavior is as if this bit is 1 other than for the purpose of a direct read.

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

**TDA, bit [9]**

Trap debug access. Traps Non-secure EL0 and EL1 System register accesses to those debug System registers in the (coproc==1110) encoding space that are not trapped by either of the following:

- [HDCR.TDRA](#).
- [HDCR.TDOSA](#).

TDA	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL0 or EL1 System register accesses to the debug registers, other than the registers trapped by <a href="#">HDCR.TDRA</a> and <a href="#">HDCR.TDOSA</a> , are trapped to Hyp mode, unless it is trapped by <a href="#">DBGDSCRext.UDCCdis</a> .

Traps of AArch32 accesses to [DBGDTRRXint](#) and [DBGDTRTXint](#) are ignored in Debug state.

If [HCR.TGE](#) or [HDCR.TDE](#) is 1, behavior is as if this bit is 1 other than for the purpose of a direct read.

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

**TDE, bit [8]**

Trap Debug exceptions. The possible values of this bit are:

TDE	Meaning
0	This control has no effect on the routing of debug exceptions, and has no effect on Non-secure accesses to debug registers.
1	In Non-secure state: <ul style="list-style-type: none"> <li>• Debug exceptions generated at EL1 or EL0 are routed to EL2.</li> <li>• The HDCR.{TDRA, TDOSA, TDA} fields are treated as being 1 for all purposes other than returning the result of a direct read of the register.</li> </ul>

When [HCR.TGE](#) == 1, the PE behaves as if the value of this field is 1 for all purposes other than returning the value of a direct read of the register.

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

**HPME, bit [7]**

Hypervisor Performance Monitors Counters Enable. The possible values of this bit are:

HPME	Meaning
0	Hyp mode Performance Monitors counters disabled.
1	Hyp mode Performance Monitors counters enabled.

When the value of this bit is 1, the Performance Monitors counters that are reserved for use from Hyp mode or Secure state are enabled. For more information see the description of the HPMN field.

If the Performance Monitors Extension is not implemented, this field is RES0.

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.

**TPM, bit [6]**

Trap Performance Monitors accesses. Traps Non-secure EL0 and EL1 accesses to all Performance Monitors registers to Hyp mode.

TPM	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL0 and EL1 accesses to all Performance Monitors registers are trapped to Hyp mode.

**Note**

EL2 does not provide traps on Performance Monitor register accesses through the optional memory-mapped external debug interface.

If the Performance Monitors Extension is not implemented, this field is RES0.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**TPMCR, bit [5]**

Trap [PMCR](#) accesses. Traps Non-secure EL0 and EL1 accesses to the [PMCR](#) to Hyp mode.

TPMCR	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL0 and EL1 accesses to the <a href="#">PMCR</a> are trapped to Hyp mode, unless it is trapped by <a href="#">PMUSERENR.EN</a> .

**Note**

EL2 does not provide traps on Performance Monitor register accesses through the optional memory-mapped external debug interface.

If the Performance Monitors Extension is not implemented, this field is RES0.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

### HPMN, bits [4:0]

Defines the number of Performance Monitors counters that are accessible from Non-secure EL1 modes, and from Non-secure EL0 modes if unprivileged access is enabled.

If the Performance Monitors Extension is not implemented, this field is RES0.

In Non-secure state, HPMN divides the Performance Monitors counters as follows. If software is accessing Performance Monitors counter *n* then, in Non-secure state:

- If *n* is in the range  $0 \leq n < \text{HPMN}$ , the counter is accessible from EL1 and EL2, and from EL0 if unprivileged access to the counters is enabled. [PMCR.E](#) enables the operation of counters in this range.
- If *n* is in the range  $\text{HPMN} \leq n < \text{PMCR.N}$ , the counter is accessible only from EL2 and from Secure state. [HDCR.HPME](#) enables the operation of counters in this range.

If this field is set to 0, or to a value larger than [PMCR.N](#), then the following CONSTRAINED UNPREDICTABLE behavior applies:

- The value returned by a direct read of [HDCR.HPMN](#) is UNKNOWN.
- Either:
  - An UNKNOWN number of counters are reserved for EL2 use. That is, the PE behaves as if [HDCR.HPMN](#) is set to an UNKNOWN non-zero value less than [PMCR.N](#).
  - All counters are reserved for EL2 use, meaning no counters are accessible from Non-secure EL1 and Non-secure EL0.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to the value of [PMCR.N](#).

## Accessing the HDCR

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>, c1, c1, 1	100	001	0001	1111	0001

## 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<sup>®</sup> 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  $\text{SCR\_EL3.NS} = 1$  &&  $\text{HCR\_EL2.E2H} = 0$  :

- 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.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDA==1, accesses to this register from EL2 are trapped to EL3 using AArch64.

02/05/2017 15:43

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

# HDFAR, Hyp Data Fault Address Register

The HDFAR characteristics are:

## Purpose

Holds the virtual address of the faulting address that caused a synchronous Data Abort exception that is taken to Hyp mode.

This register is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.

## Configuration

AArch32 System register HDFAR is architecturally mapped to AArch64 System register [FAR\\_EL2\[31:0\]](#).

AArch32 System register HDFAR is architecturally mapped to AArch32 System register [DFAR\(S\)](#) when EL2 is implemented.

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HDFAR is a 32-bit register.

## Field descriptions

The HDFAR 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 of faulting address of synchronous Data Abort exception taken to Hyp mode																															

### Bits [31:0]

VA of faulting address of synchronous Data Abort exception taken to Hyp mode.

On a Prefetch Abort exception, this register is UNKNOWN.

Any execution in a Non-secure EL1 or Non-secure EL0 mode makes this register UNKNOWN.

## Accessing the HDFAR

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>, c6, c0, 0	100	000	0110	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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T6](#)=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 [HSTR\\_EL2.T6](#)=1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T6](#)=1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# HIFAR, Hyp Instruction Fault Address Register

The HIFAR characteristics are:

## Purpose

Holds the virtual address of the faulting address that caused a synchronous Prefetch Abort exception that is taken to Hyp mode.

This register is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.

## Configuration

AArch32 System register HIFAR is architecturally mapped to AArch64 System register [FAR\\_EL2\[63:32\]](#).

AArch32 System register HIFAR is architecturally mapped to AArch32 System register [IFAR\(S\)](#) when EL2 is implemented.

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HIFAR is a 32-bit register.

## Field descriptions

The HIFAR 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 of faulting address of synchronous Prefetch Abort exception taken to Hyp mode																															

### Bits [31:0]

VA of faulting address of synchronous Prefetch Abort exception taken to Hyp mode.

On a Data Abort exception, this register is UNKNOWN.

Any execution in a Non-secure EL1 or Non-secure EL0 mode makes this register UNKNOWN.

## Accessing the HIFAR

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>, c6, c0, 2	100	010	0110	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<sup>®</sup> 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T6](#)=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 [HSTR\\_EL2.T6](#)=1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T6](#)=1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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



# HMAIR0, Hyp Memory Attribute Indirection Register 0

The HMAIR0 characteristics are:

## Purpose

Along with [HMAIR1](#), provides the memory attribute encodings corresponding to the possible AttrIdx values in a Long-descriptor format translation table entry for stage 1 translations for memory accesses from Hyp mode.

AttrIdx[2] indicates the HMAIR register to be used:

- When AttrIdx[2] is 0, HMAIR0 is used.
- When AttrIdx[2] is 1, [HMAIR1](#) is used.

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

## Configuration

AArch32 System register HMAIR0 is architecturally mapped to AArch64 System register [MAIR\\_EL2\[31:0\]](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HMAIR0 is a 32-bit register.

## Field descriptions

The HMAIR0 bit assignments are:

### When TTBCR.EAE==1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Attr3								Attr2								Attr1								Attr0							

### Attr<n>, bits [8n+7:8n], for n = 0 to 3

The memory attribute encoding for an AttrIdx[2:0] entry in a Long descriptor format translation table entry, where:

- AttrIdx[2:0] gives the value of <n> in Attr<n>.
- AttrIdx[2] defines which MAIR to access. Attr7 to Attr4 are in MAIR1, and Attr3 to Attr0 are in MAIR0.

Bits [7:4] are encoded as follows:

Attr<n>[7:4]	Meaning
0000	Device memory. See encoding of Attr<n>[3:0] for the type of Device memory.
00RW, RW not 00	Normal memory, Outer Write-Through Transient
0100	Normal memory, Outer Non-cacheable
01RW, RW not 00	Normal memory, Outer Write-Back Transient
10RW	Normal memory, Outer Write-Through Non-transient
11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

The meaning of bits [3:0] depends on the value of bits [7:4]:

Attr<n>[3:0]	Meaning when Attr<n>[7:4] is 0000	Meaning when Attr<n>[7:4] is not 0000
0000	Device-nGnRnE memory	UNPREDICTABLE
00RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Transient
0100	Device-nGnRE memory	Normal memory, Inner Non-cacheable
01RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Transient
1000	Device-nGRE memory	Normal memory, Inner Write-Through Non-transient (RW=00)
10RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Non-transient
1100	Device-GRE memory	Normal memory, Inner Write-Back Non-transient (RW=00)
11RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Non-transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in some Attr<n> fields have the following meanings:

R or W	Meaning
0	No Allocate
1	Allocate

## Accessing the HMAIR0

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>, c10, c2, 0	100	000	1010	1111	0010

## 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](#).T10==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](#).T10==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](#).T10==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# HMAIR1, Hyp Memory Attribute Indirection Register 1

The HMAIR1 characteristics are:

## Purpose

Along with [HMAIR0](#), provides the memory attribute encodings corresponding to the possible AttrIdx values in a Long-descriptor format translation table entry for stage 1 translations for memory accesses from Hyp mode.

AttrIdx[2] indicates the HMAIR register to be used:

- When AttrIdx[2] is 0, [HMAIR0](#) is used.
- When AttrIdx[2] is 1, HMAIR1 is used.

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

## Configuration

AArch32 System register HMAIR1 is architecturally mapped to AArch64 System register [MAIR\\_EL2\[63:32\]](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HMAIR1 is a 32-bit register.

## Field descriptions

The HMAIR1 bit assignments are:

### When TTBCR.EAE==1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Attr7								Attr6								Attr5								Attr4							

Attr<n>, bits [8(n-4)+7:8(n-4)], for n = 4 to 7

The memory attribute encoding for an AttrIdx[2:0] entry in a Long descriptor format translation table entry, where:

- AttrIdx[2:0] gives the value of <n> in Attr<n>.
- AttrIdx[2] defines which MAIR to access. Attr7 to Attr4 are in MAIR1, and Attr3 to Attr0 are in MAIR0.

Bits [7:4] are encoded as follows:

Attr<n>[7:4]	Meaning
0000	Device memory. See encoding of Attr<n>[3:0] for the type of Device memory.
00RW, RW not 00	Normal memory, Outer Write-Through Transient
0100	Normal memory, Outer Non-cacheable
01RW, RW not 00	Normal memory, Outer Write-Back Transient
10RW	Normal memory, Outer Write-Through Non-transient
11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

The meaning of bits [3:0] depends on the value of bits [7:4]:

Attr<n>[3:0]	Meaning when Attr<n>[7:4] is 0000	Meaning when Attr<n>[7:4] is not 0000
0000	Device-nGnRnE memory	UNPREDICTABLE
00RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Transient
0100	Device-nGnRE memory	Normal memory, Inner Non-cacheable
01RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Transient
1000	Device-nGRE memory	Normal memory, Inner Write-Through Non-transient (RW=00)
10RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Non-transient
1100	Device-GRE memory	Normal memory, Inner Write-Back Non-transient (RW=00)
11RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Non-transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in some Attr<n> fields have the following meanings:

R or W	Meaning
0	No Allocate
1	Allocate

## Accessing the HMAIR1

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>, c10, c2, 1	100	001	1010	1111	0010

## 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](#).T10==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](#).T10==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](#).T10==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# HPFAR, Hyp IPA Fault Address Register

The HPFAR characteristics are:

## Purpose

Holds the faulting IPA for some aborts on a stage 2 translation taken to Hyp mode.

This register is part of:

- The Exception and fault handling registers functional group.
- The Virtualization registers functional group.

## Configuration

AArch32 System register HPFAR is architecturally mapped to AArch64 System register [HPFAR\\_EL2\[31:0\]](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HPFAR is a 32-bit register.

## Field descriptions

The HPFAR 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
<a href="#">FIPA[39:12]</a>																												0	0	0	0

Execution in any Non-secure mode other than Hyp mode makes this register UNKNOWN.

### FIPA[39:12], bits [31:4]

Bits [39:12] of the faulting intermediate physical address.

### Bits [3:0]

Reserved, RES0.

## Accessing the HPFAR

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>, c6, c0, 4	100	100	0110	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.T6==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.T6==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.T6==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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



# HRMR, Hyp Reset Management Register

The HRMR characteristics are:

## Purpose

If EL2 is the highest implemented Exception level and this register is implemented:

- A write to the register at EL2 can request a Warm reset.
- If EL2 can use AArch32 and AArch64, this register specifies the Execution state that the PE boots into on a Warm reset.

This register is part of:

- The Virtualization registers functional group.
- The Reset management registers functional group.

## Configuration

AArch32 System register HRMR is architecturally mapped to AArch64 System register [RMR\\_EL2](#).

Only implemented if EL2 is the highest implemented Exception level. In this case:

- If EL2 can use AArch32 and AArch64 then this register must be implemented.
- If EL2 cannot use AArch64 then it is IMPLEMENTATION DEFINED whether the register is implemented.

See the field descriptions for the reset values. These apply whenever the register is implemented.

## Attributes

HRMR is a 32-bit register.

## Field descriptions

The HRMR 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	RR	AA64

### Bits [31:2]

Reserved, RES0.

### RR, bit [1]

Reset Request. Setting this bit to 1 requests a Warm reset.

This field resets to 0 on a Warm or Cold reset.

### AA64, bit [0]

When EL2 can use AArch64, determines which Execution state the PE boots into after a Warm reset:

AA64	Meaning
0	AArch32.
1	AArch64.

On coming out of the Warm reset, execution starts at the IMPLEMENTATION DEFINED reset vector address of the specified Execution state.

If EL2 cannot use AArch64 this bit is RAZ/WI.

When implemented as a RW field, this field resets to 0 on a Cold reset.

## Accessing the HRMR

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>, c12, c0, 2	100	010	1100	1111	0000

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
EL2 is the highest implemented Exception level	x	0	1	-	-	RW	n/a
EL2 is the highest implemented Exception level	x	1	1	-	n/a	RW	n/a

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

When HRMR is not implemented, the encoding for this register is UNDEFINED.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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.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.E2H==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.

# HSCTLR, Hyp System Control Register

The HSCTLR characteristics are:

## Purpose

Provides top level control of the system operation in Hyp mode.

This register is part of:

- The Virtualization registers functional group.
- The Other system control registers functional group.

## Configuration

AArch32 System register HSCTLR is architecturally mapped to AArch64 System register [SCTLR\\_EL2](#).

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 with EL2 using AArch32. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HSCTLR is a 32-bit register.

## Field descriptions

The HSCTLR 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	1	1	0	0	EE	0	1	1	0	0	WXN	1	0	1	0	0	0	I	1	0	0	SED	ITD	0	CP15BEN	LSMAOE	hT	LSMD	C	A	M

### Bit [31]

Reserved, RES0.

### TE, bit [30]

T32 Exception Enable. This bit controls whether exceptions to EL2 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 a value that is architecturally UNKNOWN.

### Bits [29:28]

Reserved, RES1.

### Bits [27:26]

Reserved, RES0.

**EE, bit [25]**

The value of the PSTATE.E bit on entry to Hyp mode, the endianness of stage 1 translation table walks in the EL2 translation regime, and the endianness of stage 2 translation table walks in the PL1&0 translation regime.

The possible values of this bit are:

EE	Meaning
0	Little-endian. PSTATE.E is cleared to 0 on entry to Hyp mode. Stage 1 translation table walks in the EL2 translation regime, and stage 2 translation table walks in the PL1&0 translation regime are little-endian.
1	Big-endian. PSTATE.E is set to 1 on entry to Hyp mode. Stage 1 translation table walks in the EL2 translation regime, and stage 2 translation table walks in the PL1&0 translation regime are big-endian.

If an implementation does not provide Big-endian support at Exception Levels higher than EL0, this bit is RES0.

If an implementation does not provide Little-endian support at Exception Levels higher than EL0, this bit is RES1.

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.

**Bits [21:20]**

Reserved, RES0.

**WXN, bit [19]**

Write permission implies XN (Execute-never). For the EL2 translation regime, this bit can force all memory regions that are writable to be treated as XN. The possible values of this bit are:

WXN	Meaning
0	This control has no effect on memory access permissions.
1	Any region that is writable in the EL2 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 [15:13]**

Reserved, RES0.

**I, bit [12]**

Instruction access Cacheability control, for accesses at EL2:

<b>I</b>	<b>Meaning</b>
0	All instruction access to Normal memory from EL2 are Non-cacheable for all levels of instruction and unified cache. If the value of HSCTLR.M is 0, instruction accesses from stage 1 of the EL2 translation regime are to Normal, Outer Shareable, Inner Non-cacheable, Outer Non-cacheable memory.
1	All instruction access to Normal memory from EL2 can be cached at all levels of instruction and unified cache. If the value of HSCTLR.M is 0, instruction accesses from stage 1 of the EL2 translation regime are to Normal, Outer Shareable, Inner Write-Through, Outer Write-Through memory.

This bit has no effect on the PL1&0 translation regime.

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 EL2.

<b>SED</b>	<b>Meaning</b>
0	SETEND instruction execution is enabled at EL2.
1	SETEND instructions are UNDEFINED at EL2.

If the implementation does not support mixed-endian operation at EL2, 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 EL2.

ITD	Meaning
0	All IT instruction functionality is enabled at EL2.
1	Any attempt at EL2 to execute any of the following is UNDEFINED: <ul style="list-style-type: none"> <li>All encodings of the IT instruction with <code>hw1[3:0] != 1000</code>.</li> <li>All encodings of the subsequent instruction with the following values for <code>hw1</code>: <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> </li> </ul> <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"> <li>A 16-bit instruction, that can only be followed by another 16-bit instruction.</li> <li>The first half of a 32-bit instruction.</li> </ul> <p>This means that, for the situations that are UNDEFINED, either the second 16-bit instruction or the 32-bit instruction is UNDEFINED. 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 HSCTLR. 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 EL2:

CP15BEN	Meaning
0	EL2 execution of the <a href="#">CP15DMB</a> , <a href="#">CP15DSB</a> , and <a href="#">CP15ISB</a> instructions is UNDEFINED.
1	EL2 execution of the <a href="#">CP15DMB</a> , <a href="#">CP15DSB</a> , and <a href="#">CP15ISB</a> instructions is enabled.

CP15BEN is optional, but if it is implemented in the [SCTLR](#) then it must also be implemented in the HSCTLR. 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.

**LSMAOE, bit [4]****In 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 EL2, A32 and T32 Load Multiple and Store Multiple can have an interrupt taken during the sequence memory accesses, and the memory accesses are not required to be ordered.
1	The ordering and interrupt behavior of A32 and T32 Load Multiple and Store Multiple at EL2 is as defined for ARMv8.0.

This bit is permitted to be cached in a TLB.

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 ARMv8.2:**

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 EL2 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 EL2 that are marked at stage 1 as Device-nGRE/Device-nGnRE/Device-nGnRnE memory are not trapped.

This bit is permitted to be cached in a TLB.

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 EL2:

C	Meaning
0	All data access to Normal memory from EL2, and all accesses to the EL2 translation tables, are Non-cacheable for all levels of data and unified cache.
1	All data access to Normal memory from EL2, and all accesses to the EL2 translation tables, can be cached at all levels of data and unified cache.

This bit has no effect on the PL1&0 translation regime.

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 or data elements 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 or data elements being accessed. If this check fails it causes an Alignment fault, which is taken as a Data Abort exception.

Load/store exclusive and load-acquire/store-release instructions have an alignment check regardless of the value of the A bit.

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 HSCTLR.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.

## Accessing the HSCTLR

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>, c1, c0, 0	100	000	0001	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<sup>®</sup> 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR](#).T1==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# HSR, Hyp Syndrome Register

The HSR characteristics are:

## Purpose

Holds syndrome information for an exception taken to Hyp mode.

This register is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.

## Configuration

AArch32 System register HSR is architecturally mapped to AArch64 System register [ESR\\_EL2](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HSR is a 32-bit register.

## Field descriptions

The HSR 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
EC						IL	ISS																								

Execution in any Non-secure PE mode other than Hyp mode makes this register UNKNOWN.

When an UNPREDICTABLE instruction is treated as UNDEFINED, and the exception is taken to EL2, the value of HSR is UNKNOWN. The value written to HSR must be consistent with a value that could be created as a result of an exception from the same Exception level that generated the exception as a result of a situation that is not UNPREDICTABLE at that Exception level, in order to avoid the possibility of a privilege violation.

### EC, bits [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about. Possible values of this field are:

EC	Meaning	ISS
000000	Unknown reason.	<a href="#">Exceptions with an unknown reason</a>
000001	Trapped WFI or WFE instruction execution. Conditional WFE and WFI instructions that fail their condition code check do not cause an exception.	<a href="#">Exception from a WFI or WFE instruction</a>
000011	Trapped MCR or MRC access with (coproc==1111) that is not reported using EC 0b000000.	<a href="#">Exception from an MCR or MRC access</a>
000100	Trapped MCRR or MRRC access with (coproc==1111) that is not reported using EC 0b000000.	<a href="#">Exception from an MCRR or MRRC access</a>
000101	Trapped MCR or MRC access with (coproc==1110).	<a href="#">Exception from an MCR or MRC access</a>
000110	Trapped LDC or STC access. The only architected uses of these instructions are: <ul style="list-style-type: none"> <li>An STC to write data to memory from <a href="#">DBGDTRRXint</a>.</li> <li>An LDC to read data from memory to <a href="#">DBGDTRTXint</a>.</li> </ul>	<a href="#">Exception from an LDC or STC instruction</a>
000111	Access to Advanced SIMD or floating-point functionality trapped by a <a href="#">HCPTR</a> . {TASE, TCP10} control. Excludes exceptions generated because Advanced SIMD and floating-point are not implemented. These are reported with EC value 0b000000.	<a href="#">Exception from an access to SIMD or floating-point functionality, resulting from HCPTR</a>
001000	Trapped VMRS access, from ID group trap, that is not reported using EC 0b000111.	<a href="#">Exception from an MCR or MRC access</a>
001100	Trapped MRRC access with (coproc==1110).	<a href="#">Exception from an MCRR or MRRC access</a>
001110	Illegal exception return to AArch32 state.	<a href="#">Exception from an Illegal state or PC alignment fault</a>
010001	Exception on SVC instruction execution in AArch32 state routed to EL2.	<a href="#">Exception from HVC or SVC instruction execution</a>
010010	HVC instruction execution in AArch32 state, when HVC is not disabled.	<a href="#">Exception from HVC or SVC instruction execution</a>
010011	Trapped execution of SMC instruction in AArch32 state.	<a href="#">Exception from SMC instruction execution</a>
100000	Prefetch Abort from a lower Exception level.	<a href="#">Exception from a Prefetch Abort</a>
100001	Prefetch Abort taken without a change in Exception level.	<a href="#">Exception from a Prefetch Abort</a>
100010	PC alignment fault exception.	<a href="#">Exception from an Illegal state or PC alignment fault</a>
100100	Data Abort from a lower Exception level.	<a href="#">Exception from a Data Abort</a>
100101	Data Abort taken without a change in Exception level.	<a href="#">Exception from a Data Abort</a>

All other EC values are reserved by ARM, and:

- Unused values in the range 0b000000 - 0b101100 (0x00 - 0x2C) are reserved for future use for synchronous exceptions.
- Unused values in the range 0b101101 - 0b111111 (0x2D - 0x3F) are reserved for future use, and might be used for synchronous or asynchronous exceptions.

The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in System and memory-mapped registers and translation table entries' in the ARM ARM, section K1.2.2.

## IL, bit [25]

Instruction length bit. Indicates the size of the instruction that has been trapped to Hyp mode. When this bit is valid, possible values of this bit are:

IL	Meaning
0	16-bit instruction trapped.
1	32-bit instruction trapped.

This field is RES1 and not valid for the following cases:

- When the EC field is 0b000000, indicating an exception with an unknown reason.
- Prefetch Aborts.
- Data Aborts for which the HSR.ISS.ISV field is 0.
- When the EC value is 0b001110, indicating an Illegal state exception.

#### Note

This is a change from the behavior in ARMv7, where the IL field is UNK/SBZP for the corresponding cases.

The IL field is not valid and is UNKNOWN on an exception from a PC alignment fault.

### ISS, bits [24:0]

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

### Exceptions with an unknown reason

This is the layout of the ISS field for exceptions with the following EC values:

- 0b000000, Unknown reason.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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

### Bits [24:0]

Reserved, RES0.

This EC code is used for all exceptions that are not covered by any other EC value. This includes exceptions that are generated in the following situations:

- The attempted execution of an instruction bit pattern that has no allocated instruction in the current PE mode in the current Security state, including:
  - A read access using a System register encoding pattern that is not allocated for reads at the current Exception level and Security state.
  - A write access using a System register encoding pattern that is not allocated for writes at the current Exception level and Security state.
  - Instruction encodings for instructions not implemented in the implementation.
- In Debug state, the attempted execution of an instruction bit pattern that is unallocated in Debug state.
- In Non-debug state, the attempted execution of an instruction bit pattern that is unallocated in Non-debug state.
- The attempted execution of a short vector floating-point instruction.
- In an implementation that does not include Advanced SIMD and floating-point functionality, an attempted access to Advanced SIMD or floating-point functionality under conditions where that access would be permitted if that functionality was present. This includes the attempted execution of an Advanced SIMD or floating-point instruction, and attempted accesses to Advanced SIMD and floating-point System registers.
- An exception generated because of the value of one of the [SCTLR](#). {ITD, SED, CP15BEN} control bits.
- Attempted execution of:
  - An HVC instruction when disabled by [HCR](#).HCD, [SCR](#).HCE, or [SCR\\_EL3](#).HCE.
  - An SMC instruction when disabled by [SCR](#).SCD or [SCR\\_EL3](#).SMD.
  - An HLT instruction when disabled by [EDSCR](#).HDE.
- An exception generated because of the attempted execution of an MSR (Banked register) or MRS (Banked register) instruction that would access a Banked register that is not accessible from the Security state and PE mode at which the instruction was executed.

#### Note

An exception is generated only if the CONSTRAINED UNPREDICTABLE behavior of the instruction is that it is UNDEFINED, see 'MSR/MRS Banked registers' in the ARMv8

ARM, section K1.1.29 (CONSTRAINED UNPREDICTABLE behavior of EL2 features).

- Attempted execution, in Debug state, of:
  - A DCPS1 instruction in Non-secure state from EL0 when EL2 is using AArch32 and the value of [HCR.TGE](#) is 1.
  - A DCPS2 instruction at EL1 or EL0 when EL2 is not implemented, or when EL3 is using AArch32 and the value of [SCR.NS](#) is 0, or when EL3 is using AArch64 and the value of [SCR\\_EL3.NS](#) is 0.
  - A DCPS3 instruction when EL3 is not implemented, or when the value of [EDSCR.SDD](#) is 1.
- In Debug state when the value of [EDSCR.SDD](#) is 1, the attempted execution at EL2, EL1, or EL0 of an instruction that is configured to trap to EL3.

'Undefined Instruction exception, when HCR.TGE is set to 1' in the ARMv8 ARM, section G1 (The AArch32 System Level Programmers' Model), describes the configuration settings for a trap that returns an HSR.EC value of 0b000000.

## Exception from a WFI or WFE instruction

This is the layout of the ISS field for exceptions with the following EC values:

- 0b000001, Trapped WFI or WFE instruction execution.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<a href="#">CV</a>	<a href="#">COND</a>				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<a href="#">TI</a>

### CV, bit [24]

Condition code valid. Possible values of this bit are:

CV	Meaning
0	The COND field is not valid.
1	The COND field is valid.

When an A32 instruction is trapped, CV is set to 1.

When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

### COND, bits [23:20]

The condition code for the trapped instruction.

When an A32 instruction is trapped, CV is set to 1 and:

- If the instruction is conditional, COND is set to the condition code field value from the instruction.
- If the instruction is unconditional, COND is set to 0b1110.

A conditional A32 instruction that is known to pass its condition code check can be presented either:

- With COND set to 0b1110, the value for unconditional.
- With the COND value held in the instruction.

When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:

- CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
- CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.

For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

### Bits [19:1]

Reserved, RES0.

**TI, bit [0]**

Trapped instruction. Possible values of this bit are:

TI	Meaning
0	WFI trapped.
1	WFE trapped.

'Trapping use of the WFI and WFE instructions' in the ARMv8 ARM, section G1 (The AArch32 System Level Programmers' Model), describes the configuration settings for this trap.

**Exception from an MCR or MRC access**

This is the layout of the ISS field for exceptions with the following EC values:

- 0b000011, Trapped MCR or MRC access with (coproc==1111) that is not reported using EC 0b000000.
- 0b000101, Trapped MCR or MRC access with (coproc==1110).
- 0b001000, Trapped VMRS access, from ID group trap, that is not reported using EC 0b000111.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				Opc2		Opc1		CRn				0		Rt			CRm			Direction			

**CV, bit [24]**

Condition code valid. Possible values of this bit are:

CV	Meaning
0	The COND field is not valid.
1	The COND field is valid.

When an A32 instruction is trapped, CV is set to 1.

When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

**COND, bits [23:20]**

The condition code for the trapped instruction.

When an A32 instruction is trapped, CV is set to 1 and:

- If the instruction is conditional, COND is set to the condition code field value from the instruction.
- If the instruction is unconditional, COND is set to 0b1110.

A conditional A32 instruction that is known to pass its condition code check can be presented either:

- With COND set to 0b1110, the value for unconditional.
- With the COND value held in the instruction.

When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:

- CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
- CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.

For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

**Opc2, bits [19:17]**

The Opc2 value from the issued instruction.

For a trapped VMRS access, holds the value 0b000.

**Opc1, bits [16:14]**

The Opc1 value from the issued instruction.

For a trapped VMRS access, holds the value 0b111.

**CRn, bits [13:10]**

The CRn value from the issued instruction.

For a trapped VMRS access, holds the reg field from the VMRS instruction encoding.

**Bit [9]**

Reserved, RES0.

**Rt, bits [8:5]**

The Rt value from the issued instruction, the general-purpose register used for the transfer.

**CRm, bits [4:1]**

The CRm value from the issued instruction.

For a trapped VMRS access, holds the value 0b0000.

**Direction, bit [0]**

Indicates the direction of the trapped instruction. The possible values of this bit are:

Direction	Meaning
0	Write to System register space. MCR instruction.
1	Read from System register space. MRC or VMRS instruction.

The following sections describe configuration settings for traps that are reported using EC value 0b000011:

- 'Traps to Hyp mode of Non-secure PL0 and PL1 accesses to the ID registers' in the ARMv8 ARM, section G1 (The AArch32 System Level Programmers' Model).
- 'Traps to Hyp mode of Non-secure PL0 and PL1 accesses to lockdown, DMA, and TCM operations' in the ARMv8 ARM, section G1.
- 'Traps to Hyp mode of Non-secure PL1 execution of cache maintenance instructions' in the ARMv8 ARM, section G1.
- 'Traps to Hyp mode of Non-secure PL1 execution of TLB maintenance instructions' in the ARMv8 ARM, section G1.
- 'Traps to Hyp mode of Non-secure PL1 accesses to the Auxiliary Control Register' in the ARMv8 ARM, section G1.
- 'Traps to Hyp mode of Non-secure PL0 and PL1 accesses to Performance Monitors registers' in the ARMv8 ARM, section G1.
- 'Traps to Hyp mode of Non-secure PL1 accesses to the CPACR' in the ARMv8 ARM, section G1.
- 'Traps to Hyp mode of Non-secure PL1 accesses to virtual memory control registers' in the ARMv8 ARM, section G1.
- 'General trapping to Hyp mode of Non-secure PL0 and PL1 accesses to CP15 System registers' in the ARMv8 ARM, section G1.

The following sections describe configuration settings for traps that are reported using EC value 0b000101:

- 'ID group 0, Primary device identification registers' in the ARMv8 ARM, section G1.
- 'Traps to Hyp mode of Non-secure CP14 accesses to trace registers' in the ARMv8 ARM, section G1.
- 'Trapping CP14 accesses to Debug ROM registers' in the ARMv8 ARM, section G1.
- 'Trapping CP14 accesses to powerdown debug registers' in the ARMv8 ARM, section G1.
- 'Trapping general CP14 accesses to debug registers' in the ARMv8 ARM, section G1.

The following sections describes configuration settings for traps that are reported using EC value 0b001000:

- 'ID group 0, Primary device identification registers' in the ARMv8 ARM, section G1.
- 'ID group 3, Detailed feature identification registers' in the ARMv8 ARM, section G1.

## Exception from an MCRR or MRRC access

This is the layout of the ISS field for exceptions with the following EC values:

- 0b000100, Trapped MCRR or MRRC access with (coproc==1111) that is not reported using EC 0b000000.
- 0b001100, Trapped MRRC access with (coproc==1110).

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				Opc1				0	0	Rt2				0	Rt				CRm			Direction	

### CV, bit [24]

Condition code valid. Possible values of this bit are:

CV	Meaning
0	The COND field is not valid.
1	The COND field is valid.

When an A32 instruction is trapped, CV is set to 1.

When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

### COND, bits [23:20]

The condition code for the trapped instruction.

When an A32 instruction is trapped, CV is set to 1 and:

- If the instruction is conditional, COND is set to the condition code field value from the instruction.
- If the instruction is unconditional, COND is set to 0b1110.

A conditional A32 instruction that is known to pass its condition code check can be presented either:

- With COND set to 0b1110, the value for unconditional.
- With the COND value held in the instruction.

When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:

- CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
- CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.

For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

### Opc1, bits [19:16]

The Opc1 value from the issued instruction.

### Bits [15:14]

Reserved, RES0.

### Rt2, bits [13:10]

The Rt2 value from the issued instruction, the second general-purpose register used for the transfer.

### Bit [9]

Reserved, RES0.



**Rt, bits [8:5]**

The Rt value from the issued instruction, the first general-purpose register used for the transfer.

**CRm, bits [4:1]**

The CRm value from the issued instruction.

**Direction, bit [0]**

Indicates the direction of the trapped instruction. The possible values of this bit are:

Direction	Meaning
0	Write to System register space. MCRR instruction.
1	Read from System register space. MRRC instruction.

The following sections describe configuration settings for traps that are reported using EC value 0b000100:

- 'Traps to Hyp mode of Non-secure PL1 accesses to virtual memory control registers' in the ARMv8 ARM, section G1 (The AArch32 System Level Programmers' Model).
- 'General trapping to Hyp mode of Non-secure PL0 and PL1 accesses to CP15 System registers' in the ARMv8 ARM, section G1.

The following sections describe configuration settings for traps that are reported using EC value 0b001100:

- 'Traps to Hyp mode of Non-secure CP14 accesses to trace registers' in the ARMv8 ARM, section G1.
- 'Trapping CP14 accesses to Debug ROM registers' in the ARMv8 ARM, section G1.

**Exception from an LDC or STC instruction**

This is the layout of the ISS field for exceptions with the following EC values:

- 0b000110, Trapped LDC or STC access.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				imm8								0	0	0	Rn			Offset	AM		Direction		

**CV, bit [24]**

Condition code valid. Possible values of this bit are:

CV	Meaning
0	The COND field is not valid.
1	The COND field is valid.

When an A32 instruction is trapped, CV is set to 1.

When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

**COND, bits [23:20]**

The condition code for the trapped instruction.

When an A32 instruction is trapped, CV is set to 1 and:

- If the instruction is conditional, COND is set to the condition code field value from the instruction.
- If the instruction is unconditional, COND is set to 0b1110.

A conditional A32 instruction that is known to pass its condition code check can be presented either:

- With COND set to 0b1110, the value for unconditional.
- With the COND value held in the instruction.

When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:

- CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
- CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.

For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

### imm8, bits [19:12]

The immediate value from the issued instruction.

### Bits [11:9]

Reserved, RES0.

### Rn, bits [8:5]

The Rn value from the issued instruction. Valid only when AM[2] is 0, indicating an immediate form of the LDC or STC instruction.

When AM[2] is 1, indicating a literal form of the LDC or STC instruction, this field is UNKNOWN.

### Offset, bit [4]

Indicates whether the offset is added or subtracted:

Offset	Meaning
0	Subtract offset.
1	Add offset.

This bit corresponds to the U bit in the instruction encoding.

### AM, bits [3:1]

Addressing mode. The permitted values of this field are:

AM	Meaning
000	Immediate unindexed.
001	Immediate post-indexed.
010	Immediate offset.
011	Immediate pre-indexed.
100	Literal unindexed. LDC instruction in A32 instruction set only. For a trapped STC instruction or a trapped T32 LDC instruction this encoding is reserved.
110	Literal offset. LDC instruction only. For a trapped STC instruction, this encoding is reserved.

The values 0b101 and 0b111 are reserved. The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

Bit [2] in this subfield indicates the instruction form, immediate or literal.

Bits [1:0] in this subfield correspond to the bits {P, W} in the instruction encoding.

### Direction, bit [0]

Indicates the direction of the trapped instruction. The possible values of this bit are:

Direction	Meaning
0	Write to memory. STC instruction.
1	Read from memory. LDC instruction.

'Trapping general Non-secure System register accesses to debug registers' in the ARMv8 ARM, section G1 describes the configuration settings for the trap that is reported using EC value 0b000110.

## Exception from an access to SIMD or floating-point functionality, resulting from HCPTR

This is the layout of the ISS field for exceptions with the following EC values:

- 0b000111, Access to Advanced SIMD or floating-point functionality trapped by a HCPTR. {TASE, TCP10} control.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV		COND			0	0	0	0	0	0	0	0	0	0	0	0	0	0	TA	0			coproc	

Excludes exceptions that occur because Advanced SIMD and floating-point functionality is not implemented, or because the value of [HCR.TGE](#) or [HCR\\_EL2.TGE](#) is 1. These are reported with EC value 0b000000.

### CV, bit [24]

Condition code valid. Possible values of this bit are:

CV	Meaning
0	The COND field is not valid.
1	The COND field is valid.

When an A32 instruction is trapped, CV is set to 1.

When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

### COND, bits [23:20]

The condition code for the trapped instruction.

When an A32 instruction is trapped, CV is set to 1 and:

- If the instruction is conditional, COND is set to the condition code field value from the instruction.
- If the instruction is unconditional, COND is set to 0b1110.

A conditional A32 instruction that is known to pass its condition code check can be presented either:

- With COND set to 0b1110, the value for unconditional.
- With the COND value held in the instruction.

When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:

- CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
- CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.

For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

### Bits [19:6]

Reserved, RES0.

### TA, bit [5]

Indicates trapped use of Advanced SIMD functionality. The possible values of this bit are:

TA	Meaning
0	Exception was not caused by trapped use of Advanced SIMD functionality.
1	Exception was caused by trapped use of Advanced SIMD functionality.

Any use of an Advanced SIMD instruction that is not also a floating-point instruction that is trapped to Hyp mode because of a trap configured in the [HCPTR](#) sets this bit to 1.

For a list of these instructions, see 'Controls of Advanced SIMD operation that do not apply to floating-point operation' in the ARMv8 ARM, section E1.

#### Bit [4]

Reserved, RES0.

#### coproc, bits [3:0]

When the TA field returns the value 1, this field returns the value 1010, otherwise this field is RES0.

The following sections describe the configuration settings for the traps that are reported using EC value 0b000111:

- 'General trapping to Hyp mode of Non-secure accesses to the SIMD and floating-point registers' in the ARMv8 ARM, section G1 (The AArch32 System Level Programmers' Model).
- 'Traps to Hyp mode of Non-secure accesses to Advanced SIMD functionality' in the ARMv8 ARM, section G1.

### Exception from HVC or SVC instruction execution

This is the layout of the ISS field for exceptions with the following EC values:

- 0b010001, Exception on SVC instruction execution in AArch32 state routed to EL2.
- 0b010010, HVC instruction execution in AArch32 state, when HVC is not disabled.

24	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	imm16																									

#### Bits [24:16]

Reserved, RES0.

#### imm16, bits [15:0]

The value of the immediate field from the HVC or SVC instruction.

For an HVC instruction, this is the value of the imm16 field of the issued instruction.

For an SVC instruction:

- If the instruction is unconditional, then:
  - For the T32 instruction, this field is zero-extended from the imm8 field of the instruction.
  - For the A32 instruction, this field is the bottom 16 bits of the imm24 field of the instruction.
- If the instruction is conditional, this field is UNKNOWN.

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

'Supervisor Call exception, when HCR.TGE is set to 1' in the ARMv8 ARM, section G1 (The AArch32 System Level Programmers' Model), describes the configuration settings for the trap reported with EC value 0b010001.

### Exception from SMC instruction execution

This is the layout of the ISS field for exceptions with the following EC values:

- 0b010011, Trapped execution of SMC instruction in AArch32 state.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND			CCKNOWNPASS				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

#### CV, bit [24]

Condition code valid. Possible values of this bit are:

CV	Meaning
0	The COND field is not valid.
1	The COND field is valid.

When an A32 instruction is trapped, CV is set to 1.

When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

This field is only valid if CCKNOWNPASS is 1, otherwise it is RES0.

### COND, bits [23:20]

The condition code for the trapped instruction.

When an A32 instruction is trapped, CV is set to 1 and:

- If the instruction is conditional, COND is set to the condition code field value from the instruction.
- If the instruction is unconditional, COND is set to 0b1110.

A conditional A32 instruction that is known to pass its condition code check can be presented either:

- With COND set to 0b1110, the value for unconditional.
- With the COND value held in the instruction.

When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:

- CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
- CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.

For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

This field is only valid if CCKNOWNPASS is 1, otherwise it is RES0.

### CCKNOWNPASS, bit [19]

Indicates whether the instruction might have failed its condition code check.

CCKNOWNPASS	Meaning
0	The instruction was unconditional, or was conditional and passed its condition code check.
1	The instruction was conditional, and might have failed its condition code check.

### Bits [18:0]

Reserved, RES0.

'Traps to Hyp mode of Non-secure PL1 execution of SMC instructions' in the ARMv8 ARM, section G1 (The AArch32 System Level Programmers' Model), describes the configuration settings for this trap, for instructions executed in Non-secure PL1 modes.

## Exception from a Prefetch Abort

This is the layout of the ISS field for exceptions with the following EC values:

- 0b100000, Prefetch Abort from a lower Exception level.
- 0b100001, Prefetch Abort taken without a change in Exception level.

24	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	FnV	EA	0	S1PTW	0					IFSC	

**Bits [24:11]**

Reserved, RES0.

**FnV, bit [10]**

FAR not Valid, for a Synchronous external abort other than a Synchronous external abort on a translation table walk.

FnV	Meaning
0	<a href="#">HIFAR</a> is valid.
1	<a href="#">HIFAR</a> is not valid, and holds an UNKNOWN value.

This field is only valid if the IFSC code is 010000. It is RES0 for all other aborts.

**EA, bit [9]**

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

**Bit [8]**

Reserved, RES0.

**S1PTW, bit [7]**

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
0	Fault not on a stage 2 translation for a stage 1 translation table walk.
1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

**Bit [6]**

Reserved, RES0.

**IFSC, bits [5:0]**

Instruction Fault Status Code. Possible values of this field are:

IFSC	Meaning
000000	Address size fault, translation table base register
000001	Address size fault, level 1
000010	Address size fault, level 2
000011	Address size fault, level 3
000101	Translation fault, level 1
000110	Translation fault, level 2
000111	Translation fault, level 3
001001	Access flag fault, level 1
001010	Access flag fault, level 2
001011	Access flag fault, level 3
001101	Permission fault, level 1
001110	Permission fault, level 2
001111	Permission fault, level 3
010000	Synchronous external abort, not on translation table walk
011000	Synchronous parity or ECC error on memory access, not on translation table walk
010101	Synchronous external abort, on translation table walk, level 1
010110	Synchronous external abort, on translation table walk, level 2
010111	Synchronous external abort, on translation table walk, level 3
011101	Synchronous parity or ECC error on memory access on translation table walk, level 1
011110	Synchronous parity or ECC error on memory access on translation table walk, level 2
011111	Synchronous parity or ECC error on memory access on translation table walk, level 3
100010	Debug exception, only when the EC value is 0b100001
110000	TLB conflict abort

All other values are reserved.

When the RAS Extension is implemented, 011000, 011101, 011110, and 011111, are reserved.

#### Note

ARMv8.2 requires the implementation of the RAS Extension.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults on a Long-descriptor translation table lookup' in the ARMv8 ARM.

The following sections describe cases where Prefetch Abort exceptions can be routed to Hyp mode, generating exceptions that are reported in the HSR with EC value 0b100000:

- 'Abort exceptions, when HCR.TGE is set to 1' in the ARMv8 ARM, section G1 (The AArch32 System Level Programmers' Model).
- 'Routing Debug exceptions to Hyp mode' in the ARMv8 ARM, section G1.

## Exception from an Illegal state or PC alignment fault

This is the layout of the ISS field for exceptions with the following EC values:

- 0b001110, Illegal exception return to AArch32 state.
- 0b100010, PC alignment fault exception.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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

#### Bits [24:0]

Reserved, RES0.

For more information about the Illegal state exception, see:

- 'Illegal changes to PSTATE.M' in the ARMv8 ARM, section G1 (The AArch32 System Level Programmers' Model).
- 'Illegal return events from AArch32 state' in the ARMv8 ARM, section G1.
- 'Legal exception returns that set PSTATE.IL to 1' in the ARMv8 ARM, section G1.

- 'The Illegal Execution state exception' in the ARMv8 ARM, section G1.

For more information about the PC alignment fault exception, see 'Branching to an unaligned PC' in the ARMv8 ARM, appendix A.

## Exception from a Data Abort

This is the layout of the ISS field for exceptions with the following EC values:

- 0b100100, Data Abort from a lower Exception level.
- 0b100101, Data Abort taken without a change in Exception level.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISV	SAS	SSE	0		SRT		0	AR	0	0	AET	FnV	EA	CM	S1	PTW	WnR				DFSC			

### ISV, bit [24]

Instruction syndrome valid. Indicates whether the syndrome information in ISS[23:14] is valid.

ISV	Meaning
0	No valid instruction syndrome. ISS[23:14] are RES0.
1	ISS[23:14] hold a valid instruction syndrome.

This bit is 0 for all faults except Data Aborts generated by stage 2 address translations for which all the following apply to the instruction that generated the Data Abort exception:

- The instruction is an LDR, LDA, LDRT, LDRSH, LDRSHT, LDRH, LDAH, LDRHT, LDRSB, LDRSBT, LDRB, LDAB, LDRBT, STR, STL, STRT, STRH, STLH, STRHT, STRB, STLB, or STRBT instruction.
- The instruction is not performing register writeback.
- The instruction is not using the PC as a source or destination register.

For these cases, ISV is UNKNOWN if the exception was generated in Debug state in memory access mode, as described in 'Data Aborts in Memory access mode' in the ARMv8 ARM, section H4.3.2 (Memory access mode), and otherwise indicates whether ISS[23:14] hold a valid syndrome.

### Note

In the A32 instruction set, LDR\*T and STR\*T instructions always perform register writeback and therefore never return a valid instruction syndrome.

When the RAS Extension is implemented, ISV is 0 for any Synchronous external abort.

ISV is set to 0 on a stage 2 abort on a stage 1 translation table walk.

When the RAS Extension is not implemented, it is IMPLEMENTATION DEFINED whether ISV is set to 1 or 0 on a Synchronous external abort on a stage 2 translation table walk.

### SAS, bits [23:22]

Syndrome Access Size. When ISV is 1, indicates the size of the access attempted by the faulting operation.

SAS	Meaning
00	Byte
01	Halfword
10	Word
11	Doubleword

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

### SSE, bit [21]

Syndrome Sign Extend. When ISV is 1, for a byte, halfword, or word load operation, indicates whether the data item must be sign extended. For these cases, the possible values of this bit are:



SSE	Meaning
0	Sign-extension not required.
1	Data item must be sign-extended.

For all other operations this bit is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

#### Bit [20]

Reserved, RES0.

#### SRT, bits [19:16]

Syndrome Register transfer. When ISV is 1, the register number of the Rt operand of the faulting instruction.

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

#### Bit [15]

Reserved, RES0.

#### AR, bit [14]

Acquire/Release. When ISV is 1, the possible values of this bit are:

AR	Meaning
0	Instruction did not have acquire/release semantics.
1	Instruction did have acquire/release semantics.

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

#### Bits [13:12]

Reserved, RES0.

#### AET, bit [11]

Asynchronous Error Type.

When the RAS Extension is implemented and the value returned in the DFSC field is 010001, describes the state of the PE after taking the SError interrupt exception. The possible values of this field are:

AET	Meaning
00	Uncontainable error (UC) or uncategorized.
01	Unrecoverable error (UEU).
10	Restartable error (UEO) or Corrected error (CE).
11	Recoverable error (UER).

On a synchronous Data Abort, this field is RES0.

If multiple errors are taken as a single SError interrupt exception, the overall state of the PE is reported. For example, if both a Recoverable and Unrecoverable error occurred, the state is Unrecoverable.

---

#### Note

---

---

Software can use this information to determine what recovery might be possible. The recovery software must also examine any implemented fault records to determine the location and extent of the error.

---

When the RAS Extension is not implemented, or when DFSC is not 010001:

- Bit[11] is RES0.
  - Bit[10] forms the FnV field.
- 

#### Note

ARMv8.2 requires the implementation of the RAS Extension.

---

### FnV, bit [10]

FAR not Valid, for a Synchronous external abort other than a Synchronous external abort on a translation table walk.

FnV	Meaning
0	<a href="#">HDFAR</a> is valid.
1	<a href="#">HDFAR</a> is not valid, and holds an UNKNOWN value.

---

When the RAS Extension is not implemented, this field is valid only if DFSC is 010000. It is RES0 for all other aborts.

When the RAS Extension is implemented:

- If DFSC is 010000, this field is valid.
  - If DFSC is 010001, this bit forms part of the AET field, becoming AET[0].
  - This field is RES0 for all other aborts.
- 

#### Note

ARMv8.2 requires the implementation of the RAS Extension.

---

### EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

### CM, bit [8]

Cache maintenance. For a synchronous fault, identifies fault that comes from a cache maintenance or address translation instruction. For synchronous faults, the possible values of this bit are:

CM	Meaning
0	Fault not generated by a cache maintenance or address translation instruction.
1	Fault generated by a cache maintenance or address translation instruction.

---

For an asynchronous Data Abort exception, this bit is 0.

### S1PTW, bit [7]

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
0	Fault not on a stage 2 translation for a stage 1 translation table walk.
1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

---

For any abort other than a stage 2 fault this bit is RES0.

**WnR, bit [6]**

Write not Read. Indicates whether a synchronous abort was caused by a write instruction or a read instruction. The possible values of this bit are:

WnR	Meaning
0	Abort caused by a read instruction.
1	Abort caused by a write instruction.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

On an asynchronous Data Abort:

- When the RAS Extension is not implemented, this bit is UNKNOWN.
- When the RAS Extension is implemented, this bit is RES0.

**Note**

ARMv8.2 requires the implementation of the RAS Extension.

**DFSC, bits [5:0]**

Data Fault Status Code. Possible values of this field are:

DFSC	Meaning
000000	Address size fault, translation table base register
000001	Address size fault, level 1
000010	Address size fault, level 2
000011	Address size fault, level 3
000101	Translation fault, level 1
000110	Translation fault, level 2
000111	Translation fault, level 3
001001	Access flag fault, level 1
001010	Access flag fault, level 2
001011	Access flag fault, level 3
001101	Permission fault, level 1
001110	Permission fault, level 2
001111	Permission fault, level 3
010000	Synchronous external abort, not on translation table walk
011000	Synchronous parity or ECC error on memory access, not on translation table walk
010001	SError interrupt
011001	SError interrupt from a parity or ECC error on memory access
010101	Synchronous external abort, on translation table walk, level 1
010110	Synchronous external abort, on translation table walk, level 2
010111	Synchronous external abort, on translation table walk, level 3
011101	Synchronous parity or ECC error on memory access on translation table walk, level 1
011110	Synchronous parity or ECC error on memory access on translation table walk, level 2
011111	Synchronous parity or ECC error on memory access on translation table walk, level 3
100001	Alignment fault
100010	Debug exception, only when the EC value is 0b100100
110000	TLB conflict abort
110100	IMPLEMENTATION DEFINED fault (Lockdown)
110101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive access)

All other values are reserved.

When the RAS Extension is implemented, 011000, 011001, 011101, 011110, and 011111, are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults on a Long-descriptor translation table lookup' in the ARMv8 ARM.

The following describe cases where Data Abort exceptions can be routed to Hyp mode, generating exceptions that are reported in the HSR with EC value 0b100100:

- 'Abort exceptions, when HCR.TGE is set to 1' in the ARMv8 ARM, section G1 (The AArch32 System Level Programmers' Model).
- 'Routing Debug exceptions to EL2' in the ARMv8 ARM, section G1.

The following describe cases that can cause a Data Abort exception that is taken to Hyp mode, and reported in the HSR with EC value of 0b100000 or 0b100100:

- 'Hyp mode control of Non-secure access permissions' in the ARMv8 ARM, section G1 (The AArch32 System Level Programmers' Model).
- 'Memory fault reporting in Hyp mode' in the ARMv8 ARM, section G1.

## Accessing the HSR

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>, c5, c2, 0	100	000	0101	1111	0010

## 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<sup>®</sup> 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.T5](#)==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.T5](#)==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.T5](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.



# HSTR, Hyp System Trap Register

The HSTR characteristics are:

## Purpose

Controls trapping to Hyp mode of Non-secure accesses, at EL1 or lower, to the System register in the coproc == 1111 encoding space, by the CRn value used to access the register using MCR or MRC instruction. When the register is accessible using an MCRR or MRRC instruction, this is the CRm value used to access the register.

This register is part of the Virtualization registers functional group.

## Configuration

AArch32 System register HSTR is architecturally mapped to AArch64 System register [HSTR\\_EL2](#).

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 with EL2 using AArch32, or into EL3 with EL3 using AArch32. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HSTR is a 32-bit register.

## Field descriptions

The HSTR 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	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0

### Bits [31:16]

Reserved, RES0.

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

Fields T14 and T4 are RES0.

The remaining fields control whether Non-secure EL0 and EL1 accesses, using MCR, MRC, MCRR, and MRRC instructions, to the System registers in the coproc == 1111 encoding space are trapped to Hyp mode:

T<n>	Meaning
0	This control has no effect on Non-secure EL0 or EL1 accesses to System registers.
1	Any Non-secure EL1 MCR, MRC access with coproc == 1111 and CRn == <n> is trapped to Hyp mode if the access is not UNDEFINED when the value of this field is 0. Any Non-secure EL1 MCRR, MRRC access with coproc == 1111 and CRm == <n> is trapped to Hyp mode if the access is not UNDEFINED when the value of this field is 0.

For example, when HSTR.T7 is 1:

- Any 32-bit access from a Non-secure EL1 mode, using an MCR or MRC instruction with coproc set to 1111 and <CRn> set to c7, and that is not UNDEFINED when HSTR.T7 is 0, is trapped to Hyp mode.
- Any 64-bit access from a Non-secure EL1 mode, using an MCRR or MRRC instruction with coproc set to 1111 and <CRm> set to c7, and that is not UNDEFINED when HSTR.T7 is 0, is trapped to Hyp mode.

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

## Accessing the HSTR

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>, c1, c1, 3	100	011	0001	1111	0001

## 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<sup>®</sup> 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.

# HTCR, Hyp Translation Control Register

The HTCR characteristics are:

## Purpose

The control register for stage 1 of the EL2 translation regime.

### Note

This stage of translation always uses the Long-descriptor translation table format.

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

## Configuration

AArch32 System register HTCR is architecturally mapped to AArch64 System register [TCR\\_EL2](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HTCR is a 32-bit register.

## Field descriptions

The HTCR 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
1	IMP DEF	0	HWU62	HWU61	HWU60	HWU59	HPD	1	0	0	0	0	0	0	0	0	0	SH0	ORGN0	IRGN0	0	0	0	0	0	0	0	0	T0SZ		

### Bit [31]

Reserved, RES1.

### IMP DEF, bit [30]

IMPLEMENTATION DEFINED.

### Bit [29]

Reserved, RES0.

### HWU62, bit [28]

In ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table block or level 3 entry if the HTCR.HPD value is 1.

Defined values are:



HWU62	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the HTCR.HPD value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

#### HWU61, bit [27]

In ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table block or level 3 entry if the HTCR.HPD value is 1.

Defined values are:

HWU61	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if HTCR.HPD bit value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

#### HWU60, bit [26]

In ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table block or level 3 entry.

Defined values are:

HWU60	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the HTCR.HPD bit value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

#### HWU59, bit [25]

In ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table block or level 3 entry.

Defined values are:

HWU59	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the HTCR.HPD bit value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

#### In ARMv8.1 and ARMv8.0:

Reserved, RES0.

#### HPD, bit [24]

##### In ARMv8.2:

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, XNTable, and PXNTable, in the PL2 translation regime.

Defined values are:

HPD	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 ARMv8.2-AA32HPD is not implemented.

#### In ARMv8.1 and ARMv8.0:

Reserved, RES0.

#### Bit [23]

Reserved, RES1.

#### Bits [22:14]

Reserved, RES0.

#### SH0, bits [13:12]

Shareability attribute for memory associated with translation table walks using [HTTBR](#).

SH0	Meaning
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 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

#### ORGN0, bits [11:10]

Outer cacheability attribute for memory associated with translation table walks using [HTTBR](#).

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 [HTTBR](#).

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 [7:3]**

Reserved, RES0.

**T0SZ, bits [2:0]**

The size offset of the memory region addressed by [HTTBR](#). The region size is  $2^{(32-T0SZ)}$  bytes.

**Accessing the HTCR**

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>, c2, c0, 2	100	010	0010	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<sup>®</sup> 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](#).T2==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](#).T2==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](#).T2==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# HTPIDR, Hyp Software Thread ID Register

The HTPIDR characteristics are:

## Purpose

Provides a location where software running in Hyp mode can store thread identifying information that is not visible to Non-secure software executing at EL0 or EL1, for hypervisor management purposes.

The PE makes no use of this register.

This register is part of:

- The Virtualization registers functional group.
- The Thread and process ID registers functional group.

## Configuration

AArch32 System register HTPIDR is architecturally mapped to AArch64 System register [TPIDR\\_EL2\[31:0\]](#).

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

The PE never updates this register. This means the register is always UNKNOWN on reset.

## Attributes

HTPIDR is a 32-bit register.

## Field descriptions

The HTPIDR 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
Thread ID																															

### Bits [31:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

## Accessing the HTPIDR

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>, c13, c0, 2	100	010	1101	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<sup>®</sup> 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](#).T13==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](#).T13==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](#).T13==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# HTTBR, Hyp Translation Table Base Register

The HTTBR characteristics are:

## Purpose

Holds the base address of the translation table for the initial lookup for stage 1 of an address translation in the EL2 translation regime, and other information for this translation regime.

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

## Configuration

AArch32 System register HTTBR is architecturally mapped to AArch64 System register [TTBR0\\_EL2](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HTTBR is a 64-bit register.

## Field descriptions

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

### Bits [63:48]

Reserved, RES0.

### BADDR, bits [47:1]

Translation table base address, bits[47:x], Bits [x-1:1] are RES0, with the additional requirement that if bits[x-1:3] are not all zero, this is a misaligned translation table base address, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Register bits [x-1:3] are treated as if all the bits are zero. The value read back from these bits is either the value written or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

x is determined from the value of [HTCR](#).T0SZ as follows:

- If [HTCR](#).T0SZ is 0 or 1,  $x = 5 - \text{HTCR.T0SZ}$ .
- If [HTCR](#).T0SZ is greater than 1,  $x = 14 - \text{HTCR.T0SZ}$ .

If bits[47:40] of the translation table base address are not zero, an Address size fault is generated.

### CnP, bit [0]

In ARMv8.2:

Common not Private. In an implementation that includes ARMv8.2-TTCNP, indicates whether each entry that is pointed to by HTTBR is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of HTTBR.CnP is 1.

CnP	Meaning
0	The translation table entries pointed to by HTTBR are permitted to differ from corresponding entries for HTTBR for other PEs in the Inner Shareable domain. This is not affected by the value of HTTBR.CnP on those other PEs.
1	The translation table entries pointed to by HTTBR are the same as the translation table entries pointed to by HTTBR on every other PE in the Inner Shareable domain for which the value of HTTBR.CnP is 1.

#### Note

If the value of the HTTBR.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those HTTBRs do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONSTRAINED UNPREDICTABLE, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values' in the ARMv8-A ARM appendix K1.

In an implementation that does not include ARMv8.2-TTCNP this field is RES0.

#### In ARMv8.1 and ARMv8.0:

Reserved, RES0.

## Accessing the HTTBR

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, 4, <Rt>, <Rt2>, c2	0100	1111	0010

## 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<sup>®</sup> 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.T2==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](#).T2==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](#).T2==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# HVBAR, Hyp Vector Base Address Register

The HVBAR characteristics are:

## Purpose

Holds the vector base address for any exception that is taken to Hyp mode.

This register is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.

## Configuration

AArch32 System register HVBAR is architecturally mapped to AArch64 System register [VBAR\\_EL2\[31:0\]](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HVBAR is a 32-bit register.

## Field descriptions

The HVBAR 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
<a href="#">Vector Base Address</a>																											0	0	0	0	0

### Bits [31:5]

Vector Base Address. Bits[31:5] of the base address of the exception vectors for exceptions taken to this Exception level. Bits[4:0] of an exception vector are the exception offset.

### Bits [4:0]

Reserved, RES0.

## Accessing the HVBAR

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>, c12, c0, 0	100	000	1100	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<sup>®</sup> 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.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.E2H==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.

02/05/2017 15:43

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

# ICC\_AP0R<n>, Interrupt Controller Active Priorities Group 0 Registers, n = 0 - 3

The ICC\_AP0R<n> characteristics are:

## Purpose

Provides information about Group 0 active priorities.

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\_AP0R<n> is architecturally mapped to AArch64 System register [ICC\\_AP0R<n>\\_EL1](#).

## Attributes

ICC\_AP0R<n> is a 32-bit register.

## Field descriptions

The ICC\_AP0R<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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

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

The contents of these registers are IMPLEMENTATION DEFINED with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

## Accessing the ICC\_AP0R<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
p15, 0, <Rt>, c12, c8, <opc2>	000	1:n<1:0>	1100	1111	1000

- <opc2> is in the range 4 - 7.

When HCR.FMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_AP0R<n>](#).

## 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	-	<a href="#">ICV_AP0R&lt;n&gt;</a>	RW	RW

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

The ICC\_AP0R<n> registers are 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\_AP0R<n> results in an access to [ICV\\_AP0R<n>](#).

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 0 active priorities) might result in UNPREDICTABLE behavior of the interrupt prioritization system, causing:

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

ICC\_AP0R1 is only implemented in implementations that support 6 or more bits of priority. ICC\_AP0R2 and ICC\_AP0R3 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

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

- ICC\_AP0R<n>.
- Secure [ICC\\_APIR<n>](#).
- Non-secure [ICC\\_APIR<n>](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# ICC\_AP1R<n>, Interrupt Controller Active Priorities Group 1 Registers, n = 0 - 3

The ICC\_AP1R<n> characteristics are:

## Purpose

Provides information about Group 1 active priorities.

This register is part of:

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

## Configuration

AArch32 System register ICC\_AP1R<n> (S) is architecturally mapped to AArch64 System register [ICC\\_AP1R<n>\\_EL1 \(S\)](#).

AArch32 System register ICC\_AP1R<n> (NS) is architecturally mapped to AArch64 System register [ICC\\_AP1R<n>\\_EL1 \(NS\)](#).

## Attributes

ICC\_AP1R<n> is a 32-bit register.

## Field descriptions

The ICC\_AP1R<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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

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

The contents of these registers are IMPLEMENTATION DEFINED with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

## Accessing the ICC\_AP1R<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
p15, 0, <Rt>, c12, c9, <opc2>	000	0:n<1:0>	1100	1111	1001

- <opc2> is in the range 0 - 3.

When HCR.IMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_AP1R<n>](#).

## 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_APIR<n>
EL3 not implemented	x	x	1	1	-	n/a	RW	n/a	ICC_APIR<n>
EL3 not implemented	x	0	0	1	-	RW	RW	n/a	ICC_APIR<n>
EL3 not implemented	x	1	0	1	-	<a href="#">ICV_APIR&lt;n&gt;</a>	RW	n/a	ICC_APIR<n>
EL3 using AArch64	x	x	1	1	-	n/a	RW	n/a	ICC_APIR<n>_ns
EL3 using AArch64	x	0	0	1	-	RW	RW	n/a	ICC_APIR<n>_ns
EL3 using AArch64	x	1	0	1	-	<a href="#">ICV_APIR&lt;n&gt;</a>	RW	n/a	ICC_APIR<n>_ns
EL3 using AArch32	x	x	1	1	-	n/a	RW	RW	ICC_APIR<n>_ns
EL3 using AArch32	x	0	0	1	-	RW	RW	RW	ICC_APIR<n>_ns
EL3 using AArch32	x	1	0	1	-	<a href="#">ICV_APIR&lt;n&gt;</a>	RW	RW	ICC_APIR<n>_ns
EL3 using AArch64	x	x	x	0	-	RW	n/a	n/a	ICC_APIR<n>_s
EL3 using AArch32	x	x	x	0	-	-	-	RW	ICC_APIR<n>_s

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

The ICC\_APIR<n> registers are 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\_APIR<n> results in an access to [ICV\\_APIR<n>](#).

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 1 active priorities) might result in UNPREDICTABLE behavior of the interrupt prioritization system, causing:

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

ICC\_APIR1 is only implemented in implementations that support 6 or more bits of priority. ICC\_APIR2 and ICC\_APIR3 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

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

- [ICC\\_AP0R<n>](#).
- Secure ICC\_APIR<n>.
- Non-secure ICC\_APIR<n>.

## 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 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# ICC\_ASGI1R, Interrupt Controller Alias Software Generated Interrupt Group 1 Register

The ICC\_ASGI1R characteristics are:

## Purpose

Generates Group 1 SGIs for the Security state that is not the current Security state.

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\_ASGI1R performs the same function as AArch64 System register [ICC\\_ASGI1R\\_EL1](#).

Under certain conditions a write to ICC\_ASGI1R can generate Group 0 interrupts, see Forwarding an SGI to a target PE.

## Attributes

ICC\_ASGI1R is a 64-bit register.

## Field descriptions

The ICC\_ASGI1R 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	Aff3								0	0	0	0	0	0	0	IRM	Aff2									
0	0	0	0	INTID				Aff1								TargetList																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

### Bits [63:56]

Reserved, RES0.

### Aff3, bits [55:48]

The affinity 3 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

### Bits [47:41]

Reserved, RES0.

### IRM, bit [40]

Interrupt Routing Mode. Determines how the generated interrupts are distributed to PEs. Possible values are:

IRM	Meaning
0	Interrupts routed to the PEs specified by Aff3.Aff2.Aff1.<target list>.
1	Interrupts routed to all PEs in the system, excluding "self".

**Aff2, bits [39:32]**

The affinity 2 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

**Bits [31:28]**

Reserved, RES0.

**INTID, bits [27:24]**

The INTID of the SGI.

**Aff1, bits [23:16]**

The affinity 1 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

**TargetList, bits [15:0]**

Target List. The set of PEs for which SGI interrupts will be generated. Each bit corresponds to the PE within a cluster with an Affinity 0 value equal to the bit number.

If a bit is 1 and the bit does not correspond to a valid target PE, the bit must be ignored by the Distributor. It is IMPLEMENTATION DEFINED whether, in such cases, a Distributor can signal a system error.

**Note**

This restricts a system to sending targeted SGIs to PEs with an affinity 0 number that is less than 16.

If SRE is set only for Secure EL3, software executing at EL3 might use the System register interface to generate SGIs. Therefore, the Distributor must always be able to receive and acknowledge Generate SGI packets received from CPU interface regardless of the ARE settings for a Security state. However, the Distributor might discard such packets.

If the IRM bit is 1, this field is RES0.

## Accessing the ICC\_ASGI1R

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	coproc	CRm
p15, 1, <Rt>, <CRn>, c12, <opc2>	0001	1111	1100

## Accessibility

The register is accessible as follows:

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

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

This register allows software executing in a Secure state to generate Non-secure Group 1 SGIs. It will also allow software executing in a Non-secure state to generate Secure Group 1 SGIs, if permitted by the settings of [GICR\\_NSACR](#) in the Redistributor corresponding to the target PE.

When [GICD\\_CTLR.DS](#)==0, Non-secure writes do not generate an interrupt for a target PE if not permitted by the [GICR\\_NSACR](#) register associated with the target PE. For more information see Use of control registers for SGI forwarding.

---

#### Note

Accesses from Secure Monitor mode are treated as Secure regardless of the value of SCR.NS.

---

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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, write accesses to this register from EL1 are UNDEFINED.

When EL2 is implemented and is using AArch64 and [SCR\\_EL3.NS](#)==1 && [.E2H](#)==0 :

- If [HCR\\_EL2.FMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR\\_EL2.IMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HSTR\\_EL2.T12](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR\\_EL3.NS](#)==1 && [.E2H](#)==1 && [HCR\\_EL2.TGE](#)==0 :

- If [HCR\\_EL2.FMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR\\_EL2.IMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HSTR\\_EL2.T12](#)==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 [HCR.FMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR.IMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HSTR.T12](#)==1, Non-secure write accesses to this register from EL1 are trapped to Hyp mode.

When [SCR\\_EL3.NS](#)==1 :

- If [ICH\\_HCR.TC](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2.TC](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch32 :

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

When EL3 is implemented and is using AArch64 and [SCR\\_EL3.NS](#)==0 :

- If [SCR\\_EL3.FIQ](#)==1, and [SCR\\_EL3.IRQ](#)==1, Secure write accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3.FIQ](#)==1, and [SCR\\_EL3.IRQ](#)==1, write 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, [SCR\\_EL3.IRQ](#)=1, [HCR.IMO](#)=0, and [HCR.FMO](#)=0, Non-secure write accesses to this register from EL1 are trapped to EL3.
- If [SCR\\_EL3.FIQ](#)=1, [SCR\\_EL3.IRQ](#)=1, [HCR\\_EL2.IMO](#)=0, and [HCR\\_EL2.FMO](#)=0, Non-secure write accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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

# ICC\_BPR0, Interrupt Controller Binary Point Register 0

The ICC\_BPR0 characteristics are:

## Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 0 interrupt preemption.

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\_BPR0 is architecturally mapped to AArch64 System register [ICC\\_BPR0\\_EL1](#).

## Attributes

ICC\_BPR0 is a 32-bit register.

## Field descriptions

The ICC\_BPR0 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	BinaryPoint

### Bits [31:3]

Reserved, RES0.

### BinaryPoint, bits [2:0]

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. This is done as follows:

Binary point value	Group priority field	Subpriority field	Field with binary point
0	[7:1]	[0]	ggggggg.s
1	[7:2]	[1:0]	gggggg.ss
2	[7:3]	[2:0]	ggggg.sss
3	[7:4]	[3:0]	gggg.ssss
4	[7:5]	[4:0]	ggg.sssss
5	[7:6]	[5:0]	gg.ssssss
6	[7]	[6:0]	g.sssssss
7	No preemption	[7:0]	.ssssssss

## Accessing the ICC\_BPR0

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 &lt;syntax&gt;

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

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c12, c8, 3	000	011	1100	1111	1000

When HCR.FMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_BPR0](#).

## 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	-	<a href="#">ICV_BPR0</a>	RW	RW

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

ICC\_BPR0 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\_BPR0 results in an access to [ICV\\_BPR0](#).

The minimum binary point value is derived from the number of implemented priority bits. The number of priority bits is IMPLEMENTATION DEFINED, and reported by [ICC\\_CTLR.PRIBits](#) and [ICC\\_MCTLR.PRIBits](#).

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value. On a reset, the binary point field is set to the minimum supported value.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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



# ICC\_BPR1, Interrupt Controller Binary Point Register 1

The ICC\_BPR1 characteristics are:

## Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption.

This register is part of:

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

## Configuration

AArch32 System register ICC\_BPR1 (S) is architecturally mapped to AArch64 System register [ICC\\_BPR1\\_EL1 \(S\)](#).

AArch32 System register ICC\_BPR1 (NS) is architecturally mapped to AArch64 System register [ICC\\_BPR1\\_EL1 \(NS\)](#).

In GIC implementations supporting two Security states, this register is Banked.

## Attributes

ICC\_BPR1 is a 32-bit register.

## Field descriptions

The ICC\_BPR1 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	BinaryPoint

### Bits [31:3]

Reserved, RES0.

### BinaryPoint, bits [2:0]

If the GIC is configured to use separate binary point fields for Group 0 and Group 1 interrupts, the value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. For more information about priorities, see Priority grouping.

Writing 0 to this field will set this field to its reset value, which is IMPLEMENTATION DEFINED and non-zero.

If EL3 is implemented and [ICC\\_MCTLR](#).CBPR\_EL1S is 1:

- Writing to this register at Secure EL1, or at EL3 not in Monitor mode, modifies [ICC\\_BPR0](#).
- Reading this register at Secure EL1, or at EL3 not in Monitor mode, returns the value of [ICC\\_BPR0](#).

If EL3 is implemented and [ICC\\_MCTLR](#).CBPR\_EL1NS is 1, Non-secure accesses to this register at EL1 or EL2 behave as follows, depending on the values of HCR.IMO and SCR.IRQ:

HCR.IMO	SCR.IRQ	Behavior
0	0	Non-secure EL1 and EL2 reads return <a href="#">ICC_BPR0</a> + 1 saturated to 0b111. Non-secure EL1 and EL2 writes are ignored.
0	1	Non-secure EL1 and EL2 accesses trap to EL3.
1	0	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 reads return <a href="#">ICC_BPR0</a> + 1 saturated to 0b111. Non-secure EL2 writes are ignored.
1	1	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 accesses trap to EL3.

If EL3 is not implemented and [ICC\\_CTLR](#).CBPR is 1, Non-secure accesses to this register at EL1 or EL2 behave as follows, depending on the values of HCR.IMO:

HCR.IMO	Behavior
0	Non-secure EL1 and EL2 reads return <a href="#">ICC_BPR0</a> + 1 saturated to 0b111. Non-secure EL1 and EL2 writes are ignored.
1	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 reads return <a href="#">ICC_BPR0</a> + 1 saturated to 0b111. Non-secure EL2 writes are ignored.

## Accessing the ICC\_BPR1

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, 3	000	011	1100	1111	1100

When HCR.IMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_BPR1](#).

## 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_BPR1
EL3 not implemented	x	x	1	1	-	n/a	RW	n/a	ICC_BPR1
EL3 not implemented	x	0	0	1	-	RW	RW	n/a	ICC_BPR1
EL3 not implemented	x	1	0	1	-	<a href="#">ICV_BPR1</a>	RW	n/a	ICC_BPR1
EL3 using AArch64	x	x	1	1	-	n/a	RW	n/a	ICC_BPR1_ns
EL3 using AArch64	x	0	0	1	-	RW	RW	n/a	ICC_BPR1_ns
EL3 using AArch64	x	1	0	1	-	<a href="#">ICV_BPR1</a>	RW	n/a	ICC_BPR1_ns
EL3 using AArch32	x	x	1	1	-	n/a	RW	RW	ICC_BPR1_ns
EL3 using AArch32	x	0	0	1	-	RW	RW	RW	ICC_BPR1_ns
EL3 using AArch32	x	1	0	1	-	<a href="#">ICV_BPR1</a>	RW	RW	ICC_BPR1_ns
EL3 using AArch64	x	x	x	0	-	RW	n/a	n/a	ICC_BPR1_s
EL3 using AArch32	x	x	x	0	-	-	-	RW	ICC_BPR1_s

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

ICC\_BPR1 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\_BPR1 results in an access to [ICV\\_BPR1](#).

---

When the PE resets into an Exception level that is using AArch32, the reset value is equal to:

- For the Secure copy of the register, the minimum value of [ICC\\_BPR0](#) plus one.
- For the Non-secure copy of the register, the minimum value of [ICC\\_BPR0](#).

Where the minimum value of [ICC\\_BPR0](#) is IMPLEMENTATION DEFINED.

If EL3 is not implemented:

- If the PE is Secure this reset value is (minimum value of [ICC\\_BPR0](#) plus one).
- If the PE is Non-secure this reset value is (minimum value of [ICC\\_BPR0](#)).

An attempt to program the binary point field to a value less than the reset value sets the field to the reset value.

## 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 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# ICC\_CTLR, Interrupt Controller Control Register

The ICC\_CTLR characteristics are:

## Purpose

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

This register is part of:

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

## Configuration

AArch32 System register ICC\_CTLR (S) is architecturally mapped to AArch64 System register [ICC\\_CTLR\\_EL1 \(S\)](#).

AArch32 System register ICC\_CTLR (NS) is architecturally mapped to AArch64 System register [ICC\\_CTLR\\_EL1 \(NS\)](#).

## Attributes

ICC\_CTLR is a 32-bit register.

## Field descriptions

The ICC\_CTLR 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	<a href="#">A3V</a>	<a href="#">SEIS</a>	<a href="#">IDbits</a>	<a href="#">PRIbits</a>	0	<a href="#">PMHE</a>	0	0	0	0	<a href="#">EOImode</a>	<a href="#">CBPR</a>				

### Bits [31:16]

Reserved, RES0.

### A3V, bit [15]

Affinity 3 Valid. Read-only and writes are ignored. Possible values are:

A3V	Meaning
0	The CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers.
1	The CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

If EL3 is implemented and using AArch32, this bit is an alias of [ICC\\_MCTLR](#).A3V.

If EL3 is implemented and using AArch64, this bit is an alias of [ICC\\_CTLR\\_EL3](#).A3V.

### SEIS, bit [14]

SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs:

SEIS	Meaning
0	The CPU interface logic does not support local generation of SEIs.
1	The CPU interface logic supports local generation of SEIs.

If EL3 is implemented and using AArch32, this bit is an alias of [ICC\\_MCTLR](#).SEIS.

If EL3 is implemented and using AArch64, this bit is an alias of [ICC\\_CTLR\\_EL3](#).SEIS.

### IDbits, bits [13:11]

Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported:

IDbits	Meaning
000	16 bits.
001	24 bits.

All other values are reserved.

If EL3 is implemented and using AArch32, this field is an alias of [ICC\\_MCTLR](#).IDbits.

If EL3 is implemented and using AArch64, this field is an alias of [ICC\\_CTLR\\_EL3](#).IDbits.

### PRbits, bits [10:8]

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).

An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).

---

#### Note

This field always returns the number of priority bits implemented, regardless of the Security state of the access or the value of [GICD\\_CTLR](#).DS.

---

The division between group priority and subpriority is defined in the binary point registers [ICC\\_BPR0](#) and [ICC\\_BPR1](#).

If EL3 is implemented and using AArch32, physical accesses return the value from [ICC\\_MCTLR](#).PRbits.

If EL3 is implemented and using AArch64, physical accesses return the value from [ICC\\_CTLR\\_EL3](#).PRbits.

If EL3 is not implemented, physical accesses return the value from this field.

### Bit [7]

Reserved, RES0.

### PMHE, bit [6]

Priority Mask Hint Enable. Controls whether the priority mask register is used as a hint for interrupt distribution:

PMHE	Meaning
0	Disables use of <a href="#">ICC_PMR</a> as a hint for interrupt distribution.
1	Enables use of <a href="#">ICC_PMR</a> as a hint for interrupt distribution.

If EL3 is implemented:

- If EL3 is using AArch32, this bit is an alias of [ICC\\_MCTLR](#).PMHE.
- If EL3 is using AArch64, this bit is an alias of [ICC\\_CTLR\\_EL3](#).PMHE.
- If [GICD\\_CTLR](#).DS == 0, this bit is read-only.
- If [GICD\\_CTLR](#).DS == 1, this bit is read/write.

If EL3 is not implemented, it is IMPLEMENTATION DEFINED whether this bit is read-only or read-write:

- If this bit is read-only, an implementation can choose to make this field RAZ/WI or RAO/WI.
- If this bit is read/write, it resets to zero.

### Bits [5:2]

Reserved, RES0.

**EOImode, bit [1]**

EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt:

EOImode	Meaning
0	<a href="#">ICC_EOIR0</a> and <a href="#">ICC_EOIR1</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">ICC_DIR</a> are UNPREDICTABLE.
1	<a href="#">ICC_EOIR0</a> and <a href="#">ICC_EOIR1</a> provide priority drop functionality only. <a href="#">ICC_DIR</a> provides interrupt deactivation functionality.

If EL3 is implemented:

- If EL3 is using AArch32, this bit is an alias of [ICC\\_MCTLR.EOImode\\_EL1](#) {S, NS} where S or NS corresponds to the current Security state.
- If EL3 is using AArch64, this bit is an alias of [ICC\\_CTLR\\_EL3.EOImode\\_EL1](#) {S, NS} where S or NS corresponds to the current Security state.

If EL3 is not implemented, it is IMPLEMENTATION DEFINED whether this bit is read-only or read-write:

- If this bit is read-only, an implementation can choose to make this field RAZ/WI or RAO/WI.
- If this bit is read/write, it resets to zero.

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.

**CBPR, bit [0]**

Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupts:

CBPR	Meaning
0	<a href="#">ICC_BPR0</a> determines the preemption group for Group 0 interrupts only. <a href="#">ICC_BPR1</a> determines the preemption group for Group 1 interrupts.
1	<a href="#">ICC_BPR0</a> determines the preemption group for both Group 0 and Group 1 interrupts.

If EL3 is implemented:

- If EL3 is using AArch32, this bit is an alias of [ICC\\_MCTLR.CBPR\\_EL1](#) {S,NS} where S or NS corresponds to the current Security state.
- If EL3 is using AArch64, this bit is an alias of [ICC\\_CTLR\\_EL3.CBPR\\_EL1](#) {S,NS} where S or NS corresponds to the current Security state.
- If [GICD\\_CTLR.DS](#) == 0, this bit is read-only.
- If [GICD\\_CTLR.DS](#) == 1, this bit is read/write.

If EL3 is not implemented, it is IMPLEMENTATION DEFINED whether this bit is read-only or read-write:

- If this bit is read-only, an implementation can choose to make this field RAZ/WI or RAO/WI.
- If this bit is read/write, it resets to zero.

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

**Accessing the ICC\_CTLR**

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, 4	000	100	1100	1111	1100

When HCR.{FMO, IMO} != {0, 0}, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_CTLR](#).

## 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_CTLR
EL3 not implemented	x	x	1	1	-	n/a	RW	n/a	ICC_CTLR
EL3 not implemented	x	1	0	1	-	<a href="#">ICV_CTLR</a>	RW	n/a	ICC_CTLR
EL3 not implemented	1	x	0	1	-	<a href="#">ICV_CTLR</a>	RW	n/a	ICC_CTLR
EL3 not implemented	0	0	0	1	-	RW	RW	n/a	ICC_CTLR
EL3 using AArch64	x	x	x	0	-	RW	n/a	n/a	ICC_CTLR_s
EL3 using AArch32	x	x	x	0	-	-	-	RW	ICC_CTLR_s
EL3 using AArch64	x	x	1	1	-	n/a	RW	n/a	ICC_CTLR_ns
EL3 using AArch64	x	1	0	1	-	<a href="#">ICV_CTLR</a>	RW	n/a	ICC_CTLR_ns
EL3 using AArch64	1	x	0	1	-	<a href="#">ICV_CTLR</a>	RW	n/a	ICC_CTLR_ns
EL3 using AArch64	0	0	0	1	-	RW	RW	n/a	ICC_CTLR_ns
EL3 using AArch32	x	x	1	1	-	n/a	RW	RW	ICC_CTLR_ns
EL3 using AArch32	x	1	0	1	-	<a href="#">ICV_CTLR</a>	RW	RW	ICC_CTLR_ns
EL3 using AArch32	1	x	0	1	-	<a href="#">ICV_CTLR</a>	RW	RW	ICC_CTLR_ns
EL3 using AArch32	0	0	0	1	-	RW	RW	RW	ICC_CTLR_ns

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

ICC\_CTLR is only accessible at Non-secure EL1 when  $HCR.\{FMO, IMO\} == \{0, 0\}$ .

### Note

When  $HCR.\{FMO, IMO\} != \{0, 0\}$ , at Non-secure EL1, the instruction encoding used to access ICC\_CTLR results in an access to [ICV\\_CTLR](#).

## 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  $SCR\_EL3.NS==1 \ \&\& \ .E2H==0$  :

- If [HSTR\\_EL2](#).T12==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and  $SCR\_EL3.NS==1 \ \&\& \ .E2H==1 \ \&\& \ HCR\_EL2.TGE==0$  :

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



- If [ICH\\_HCR\\_EL2](#).TC==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, and [SCR](#).FIQ==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, [SCR](#).FIQ==1, [HCR](#).IMO==0, and [HCR](#).FMO==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](#).FIQ==1, and [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](#).FIQ==1, and [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](#).FIQ==1, [SCR\\_EL3](#).IRQ==1, [HCR](#).IMO==0, and [HCR](#).FMO==0, Non-secure accesses to this register from EL1 are trapped to EL3.
- If [SCR\\_EL3](#).FIQ==1, [SCR\\_EL3](#).IRQ==1, [HCR\\_EL2](#).IMO==0, and [HCR\\_EL2](#).FMO==0, Non-secure accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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

# ICC\_DIR, Interrupt Controller Deactivate Interrupt Register

The ICC\_DIR characteristics are:

## Purpose

When interrupt priority drop is separated from interrupt deactivation, a write to this register deactivates the specified interrupt.

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\_DIR performs the same function as AArch64 System register [ICC\\_DIR\\_EL1](#).

## Attributes

ICC\_DIR is a 32-bit register.

## Field descriptions

The ICC\_DIR 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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the interrupt to be deactivated.

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

## Accessing the ICC\_DIR

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, c11, 1	000	001	1100	1111	1011

This encoding results in an access to [ICV\\_DIR](#) at Non-secure EL1 in the following cases:

- When HCR.FMO is set to 1.
- When HCR.IMO is set to 1.

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	WO	n/a	WO
x	x	1	1	-	n/a	WO	WO
x	1	0	1	-	<a href="#">ICV_DIR</a>	WO	WO
1	x	0	1	-	<a href="#">ICV_DIR</a>	WO	WO
0	0	0	1	-	WO	WO	WO

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

The ICC\_DIR register is only accessible at Non-secure EL1 when  $HCR.\{FMO, IMO\} == \{0, 0\}$ .

### Note

At Non-secure EL1, the instruction encoding used to access ICC\_DIR results in an access to [ICV\\_DIR](#) in the following cases:

- When HCR.FMO is set to 1.
- When HCR.IMO is set to 1.

There are two cases when writing to [ICC\\_DIR\\_EL1](#) that were UNPREDICTABLE for a corresponding GICv2 write to [GICC\\_DIR](#):

- When EOImode == '0'. GICv3 implementations must ignore such writes. In systems supporting system error generation, an implementation might generate an SEI.
- When EOImode == '1' but no EOI has been issued. The interrupt will be de-activated by the Distributor, however the active priority in the CPU interface for the interrupt will remain set (because no EOI was issued).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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, write accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_HSRE](#).SRE==0, write accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, write accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and  $SCR\_EL3.NS==1 \ \&\& \ .E2H==0$  :

- If [HSTR\\_EL2](#).T12==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and  $SCR\_EL3.NS==1 \ \&\& \ .E2H==1 \ \&\& \ HCR\_EL2.TGE==0$  :

- If [HSTR\\_EL2](#).T12==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 [HSTR](#).T12==1, Non-secure write accesses to this register from EL1 are trapped to Hyp mode.

When  $SCR\_EL3.NS==1$  :

- If [ICH\\_HCR](#).TC==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch32 :

- If [SCR.IRQ](#)==1, and [SCR.FIQ](#)==1, write 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, [SCR.FIQ](#)==1, [HCR.IMO](#)==0, and [HCR.FMO](#)==0, Non-secure write 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 [SCR\\_EL3.IRQ](#)==1, Secure write accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3.FIQ](#)==1, and [SCR\\_EL3.IRQ](#)==1, write 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, [SCR\\_EL3.IRQ](#)==1, [HCR.IMO](#)==0, and [HCR.FMO](#)==0, Non-secure write accesses to this register from EL1 are trapped to EL3.
- If [SCR\\_EL3.FIQ](#)==1, [SCR\\_EL3.IRQ](#)==1, [HCR\\_EL2.IMO](#)==0, and [HCR\\_EL2.FMO](#)==0, Non-secure write accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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

# ICC\_EOIR0, Interrupt Controller End Of Interrupt Register 0

The ICC\_EOIR0 characteristics are:

## Purpose

A PE writes to this register to inform the CPU interface that it has completed the processing of the specified Group 0 interrupt.

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\_EOIR0 performs the same function as AArch64 System register [ICC\\_EOIR0\\_EL1](#).

## Attributes

ICC\_EOIR0 is a 32-bit register.

## Field descriptions

The ICC\_EOIR0 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	INTID																										

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID from the corresponding [ICC\\_IAR0](#) access.

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

If the EOImode bit for the current Exception level and Security state is 0, a write to this register drops the priority for the interrupt, and also deactivates the interrupt.

If the EOImode bit for the current Exception level and Security state is 1, a write to this register only drops the priority for the interrupt. Software must write to [ICC\\_DIR](#) to deactivate the interrupt.

The appropriate EOImode bit varies as follows:

- If EL3 is not implemented, the appropriate bit is [ICC\\_CTLR.EOImode](#).
- If EL3 is implemented and the software is executing in Monitor mode, the appropriate bit is [ICC\\_MCTLR.EOImode\\_EL3](#).
- If EL3 is implemented and the software is not executing in Monitor mode, the bit depends on the current Security state:
  - If the software is executing in Secure state, the bit is [ICC\\_CTLR.EOImode](#) in the Secure instance of [ICC\\_CTLR](#). This is an alias of [ICC\\_MCTLR.EOImode\\_EL1S](#).
  - If the software is executing in Non-secure state, the bit is [ICC\\_CTLR.EOImode](#) in the Non-secure instance of [ICC\\_CTLR](#). This is an alias of [ICC\\_MCTLR.EOImode\\_EL1NS](#).

## Accessing the ICC\_EOIR0

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, c8, 1	000	001	1100	1111	1000

When HCR.FMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_EOIR0](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	WO	n/a	WO
x	x	1	1	-	n/a	WO	WO
0	x	0	1	-	WO	WO	WO
1	x	0	1	-	<a href="#">ICV_EOIR0</a>	WO	WO

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

ICC\_EOIR0 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\_EOIR0 results in an access to [ICV\\_EOIR0](#).

A write to this register must correspond to the most recent valid read by this PE from an Interrupt Acknowledge Register, and must correspond to the INTID that was read from [ICC\\_IAR0](#), otherwise the system behavior is UNPREDICTABLE. A valid read is a read that returns a valid INTID that is not a special INTID.

A write of a Special INTID is ignored. See Special INTIDs, for more information.

## 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, write accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_HSRE](#).SRE==0, write accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, write accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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 [HSTR.T12](#)==1, Non-secure write accesses to this register from EL1 are trapped to Hyp mode.

When [SCR\\_EL3.NS](#)==1 :

- If [ICH\\_HCR.TALL0](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2.TALL0](#)==1, Non-secure write 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, write 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 write 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 write 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, write 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 write 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 write accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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

# ICC\_EOIR1, Interrupt Controller End Of Interrupt Register 1

The ICC\_EOIR1 characteristics are:

## Purpose

A PE writes to this register to inform the CPU interface that it has completed the processing of the specified Group 1 interrupt.

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\_EOIR1 performs the same function as AArch64 System register [ICC\\_EOIR1\\_EL1](#).

## Attributes

ICC\_EOIR1 is a 32-bit register.

## Field descriptions

The ICC\_EOIR1 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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID from the corresponding [ICC\\_IAR1](#) access.

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

If the EOImode bit for the current Exception level and Security state is 0, a write to this register drops the priority for the interrupt, and also deactivates the interrupt.

If the EOImode bit for the current Exception level and Security state is 1, a write to this register only drops the priority for the interrupt. Software must write to [ICC\\_DIR](#) to deactivate the interrupt.

The appropriate EOImode bit varies as follows:

- If EL3 is not implemented, the appropriate bit is [ICC\\_CTLR](#).EOImode.
- If EL3 is implemented and the software is executing in Monitor mode, the appropriate bit is [ICC\\_MCTLR](#).EOImode\_EL3.
- If EL3 is implemented and the software is not executing in Monitor mode, the bit depends on the current Security state:
  - If the software is executing in Secure state, the bit is [ICC\\_CTLR](#).EOImode in the Secure instance of [ICC\\_CTLR](#). This is an alias of [ICC\\_MCTLR](#).EOImode\_EL1S.
  - If the software is executing in Non-secure state, the bit is [ICC\\_CTLR](#).EOImode in the Non-secure instance of [ICC\\_CTLR](#). This is an alias of [ICC\\_MCTLR](#).EOImode\_EL1NS.



## Accessing the ICC\_EOIR1

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, 1	000	001	1100	1111	1100

When HCR.IMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_EOIR1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	WO	n/a	WO
x	x	1	1	-	n/a	WO	WO
x	0	0	1	-	WO	WO	WO
x	1	0	1	-	<a href="#">ICV_EOIR1</a>	WO	WO

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

ICC\_EOIR1 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\_EOIR1 results in an access to [ICV\\_EOIR1](#).

A write to this register must correspond to the most recent valid read by this PE from an Interrupt Acknowledge Register, and must correspond to the INTID that was read from [ICC\\_IARI](#), otherwise the system behavior is UNPREDICTABLE. A valid read is a read that returns a valid INTID that is not a special INTID.

A write of a Special INTID is ignored. See Special INTIDs, for more information.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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, write accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_HSRE](#).SRE==0, write accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, write accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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 [HSTR.T12](#)==1, Non-secure write accesses to this register from EL1 are trapped to Hyp mode.

When [SCR\\_EL3.NS](#)==1 :

- If [ICH\\_HCR.TALL1](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2.TALL1](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch32 :

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

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3.IRQ](#)==1, write 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 write accesses to this register from EL1 are trapped to EL3.
- If [SCR\\_EL3.IRQ](#)==1, and [HCR\\_EL2.IMO](#)==0, Non-secure write accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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

# ICC\_HPPIR0, Interrupt Controller Highest Priority Pending Interrupt Register 0

The ICC\_HPPIR0 characteristics are:

## Purpose

Indicates the highest priority pending Group 0 interrupt on the CPU interface.

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\_HPPIR0 performs the same function as AArch64 System register [ICC\\_HPPIR0\\_EL1](#).

## Attributes

ICC\_HPPIR0 is a 32-bit register.

## Field descriptions

The ICC\_HPPIR0 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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the highest priority pending interrupt, if that interrupt is observable at the current Security state and Exception level.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. These special INTIDs can be one of: 1020, 1021, or 1023. See Special INTIDs, for more information.

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

## Accessing the ICC\_HPPIR0

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>, c12, c8, 2	000	010	1100	1111	1000

When HCR.FMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_HPPIR0](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	RO	n/a	RO
x	x	1	1	-	n/a	RO	RO
0	x	0	1	-	RO	RO	RO
1	x	0	1	-	<a href="#">ICV_HPPIR0</a>	RO	RO

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

ICC\_HPPIR0 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\_HPPIR0 results in an access to [ICV\\_HPPIR0](#).

## 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, read accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_HSRE](#).SRE==0, read accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, read accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==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 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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](#).T12==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR](#).TALL0==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure read 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, read 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 read 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 read 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, read 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 read 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 read accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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

# ICC\_HPPIR1, Interrupt Controller Highest Priority Pending Interrupt Register 1

The ICC\_HPPIR1 characteristics are:

## Purpose

Indicates the highest priority pending Group 1 interrupt on the CPU interface.

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\_HPPIR1 performs the same function as AArch64 System register [ICC\\_HPPIR1\\_EL1](#).

## Attributes

ICC\_HPPIR1 is a 32-bit register.

## Field descriptions

The ICC\_HPPIR1 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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the highest priority pending interrupt, if that interrupt is observable at the current Security state and Exception level.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. This special INTID can take the value 1023 only. See Special INTIDs, for more information.

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

## Accessing the ICC\_HPPIR1

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>, c12, c12, 2	000	010	1100	1111	1100

When HCR.IMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_HPPIR1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	RO	n/a	RO
x	x	1	1	-	n/a	RO	RO
x	0	0	1	-	RO	RO	RO
x	1	0	1	-	<a href="#">ICV_HPPIR1</a>	RO	RO

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

ICC\_HPPIR1 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\_HPPIR1 results in an access to [ICV\\_HPPIR1](#).

## 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, read accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_HSRE](#).SRE==0, read accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, read accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==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 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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](#).T12==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR](#).TALL1==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch32 :

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

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).IRQ==1, read 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 read accesses to this register from EL1 are trapped to EL3.
- If [SCR\\_EL3](#).IRQ==1, and [HCR\\_EL2](#).IMO==0, Non-secure read accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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



# ICC\_HSRE, Interrupt Controller Hyp System Register Enable register

The ICC\_HSRE characteristics are:

## Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL2.

This register is part of:

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

## Configuration

AArch32 System register ICC\_HSRE is architecturally mapped to AArch64 System register [ICC\\_SRE\\_EL2](#).

## Attributes

ICC\_HSRE is a 32-bit register.

## Field descriptions

The ICC\_HSRE 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	Enable	DIB	DFB	SRE

### Bits [31:4]

Reserved, RES0.

### Enable, bit [3]

Enable. Enables lower Exception level access to [ICC\\_SRE](#).

Enable	Meaning
0	Non-secure EL1 accesses to <a href="#">ICC_SRE</a> trap to EL2.
1	Non-secure EL1 accesses to <a href="#">ICC_SRE</a> do not trap to EL2.

If ICC\_HSRE.SRE is RAO/WI, an implementation is permitted to make the Enable bit RAO/WI.

If ICC\_HSRE.SRE is 0, the Enable bit behaves as 1 for all purposes other than reading the value of the bit.

### DIB, bit [2]

Disable IRQ bypass.

DIB	Meaning
0	IRQ bypass enabled.
1	IRQ bypass disabled.

If EL3 is implemented and [GICD\\_CTLR](#).DS is 0, this field is a read-only alias of [ICC\\_MSRE](#).DIB.

If EL3 is implemented and [GICD\\_CTLR](#).DS is 1, this field is a read-write alias of [ICC\\_MSRE](#).DIB.

In systems that do not support IRQ bypass, 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 0.

### DFB, bit [1]

Disable FIQ bypass.

DFB	Meaning
0	FIQ bypass enabled.
1	FIQ bypass disabled.

If EL3 is implemented and [GICD\\_CTLR.DS](#) is 0, this field is a read-only alias of [ICC\\_MSRE.DFB](#).

If EL3 is implemented and [GICD\\_CTLR.DS](#) is 1, this field is a read-write alias of [ICC\\_MSRE.DFB](#).

In systems that do not support FIQ bypass, 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 0.

### SRE, bit [0]

System Register Enable.

SRE	Meaning
0	The memory-mapped interface must be used. Accesses at EL2 or below to any ICH_* System register, or any EL1 or EL2 ICC_* register other than <a href="#">ICC_SRE</a> or <a href="#">ICC_HSRE</a> , are UNDEFINED.
1	The System register interface to the ICH_* registers and the EL1 and EL2 ICC_* registers is enabled for EL2.

If software changes this bit from 1 to 0, the results are UNPREDICTABLE.

If an implementation supports only a System register interface to the GIC CPU interface, this bit is RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

## Accessing the ICC\_HSRE

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>, c12, c9, 5	100	101	1100	1111	1001

## Accessibility

The register is accessible as follows:

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

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

The GIC architecture permits, but does not require, that registers can be shared between memory-mapped registers and the equivalent System registers. This means that if the memory-mapped registers have been accessed while `ICC_HSRE.SRE==0`, then the System registers might be modified. Therefore, software must only rely on the reset values of the System registers if there has been no use of the GIC functionality while the memory-mapped registers are in use. Otherwise, the System register values must be treated as UNKNOWN.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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_MSRE.Enable==0`, accesses to this register from EL2 are UNDEFINED.
- If `ICC_SRE_EL3.Enable==0`, accesses to this register from EL2 are trapped to EL3.

When EL2 is implemented and is using AArch64 and `SCR_EL3.NS==1 && .E2H==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 AArch64 and `SCR_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# ICC\_IAR0, Interrupt Controller Interrupt Acknowledge Register 0

The ICC\_IAR0 characteristics are:

## Purpose

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

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\_IAR0 performs the same function as AArch64 System register [ICC\\_IAR0\\_EL1](#).

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

## Attributes

ICC\_IAR0 is a 32-bit register.

## Field descriptions

The ICC\_IAR0 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																								

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the signaled interrupt.

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

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. These special INTIDs can be one of: 1020, 1021, or 1023. See Special INTIDs, for more information.

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

## Accessing the ICC\_IAR0

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>, c12, c8, 0	000	000	1100	1111	1000

When HCR.FMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_IAR0](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	RO	n/a	RO
x	x	1	1	-	n/a	RO	RO
0	x	0	1	-	RO	RO	RO
1	x	0	1	-	<a href="#">ICV_IAR0</a>	RO	RO

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

ICC\_IAR0 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\_IAR0 results in an access to [ICV\\_IAR0](#).

## 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, read accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_HSRE](#).SRE==0, read accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, read accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==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 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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](#).T12==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When SCR\_EL3.NS==1 :

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

- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure read 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, read 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 read 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 read 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, read 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 read 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 read accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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

# ICC\_IAR1, Interrupt Controller Interrupt Acknowledge Register 1

The ICC\_IAR1 characteristics are:

## Purpose

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

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\_IAR1 performs the same function as AArch64 System register [ICC\\_IAR1\\_EL1](#).

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

## Attributes

ICC\_IAR1 is a 32-bit register.

## Field descriptions

The ICC\_IAR1 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																								

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the signaled interrupt.

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

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. This special INTID can take the value 1023 only. See Special INTIDs, for more information.

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

## Accessing the ICC\_IAR1

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>, c12, c12, 0	000	000	1100	1111	1100

When HCR.IMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_IAR1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	RO	n/a	RO
x	x	1	1	-	n/a	RO	RO
x	0	0	1	-	RO	RO	RO
x	1	0	1	-	<a href="#">ICV_IAR1</a>	RO	RO

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

ICC\_IAR1 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\_IAR1 results in an access to [ICV\\_IAR1](#).

## 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, read accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_HSRE](#).SRE==0, read accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, read accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==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 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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](#).T12==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When SCR\_EL3.NS==1 :

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



- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch32 :

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

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).IRQ==1, read 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 read accesses to this register from EL1 are trapped to EL3.
- If [SCR\\_EL3](#).IRQ==1, and [HCR\\_EL2](#).IMO==0, Non-secure read accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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

# 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:

<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	-	<a href="#">ICV_IGRPEN0</a>	RW	RW

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<sup>®</sup> 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 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# 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	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 &lt;syntax&gt;

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

MCR &lt;syntax&gt;

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:

<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	-	<a href="#">ICV_IGRPEN1</a>	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	-	<a href="#">ICV_IGRPEN1</a>	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>, c12, c12, 7	EL3 using AArch32	x	1	0	1	-	<a href="#">ICV_IGRPEN1</a>	RW	RW	ICC_IGRPEN1_ns
p15, 0, <Rt>, c12, c12, 7	EL3 using AArch64	x	x	x	0	-	RW	n/a	n/a	ICC_IGRPEN1_s
p15, 0, <Rt>, c12, c12, 7	EL3 using AArch32	x	x	x	0	-	-	-	RW	ICC_IGRPEN1_s

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 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

# ICC\_MCTLR, Interrupt Controller Monitor Control Register

The ICC\_MCTLR characteristics are:

## Purpose

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

This register is part of:

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

## Configuration

This register is only accessible in Secure state.

AArch32 System register ICC\_MCTLR can be mapped to AArch64 System register [ICC\\_CTLR\\_EL3](#), but this is not architecturally mandated.

## Attributes

ICC\_MCTLR is a 32-bit register.

## Field descriptions

The ICC\_MCTLR 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	nDS	0	A3V	SEIS	IDbits	PRIBits	0	PMHE	RM	EOImode_EL1NS	EOImode_EL1S	EOImode_EL3	CBPR_E				

### Bits [31:18]

Reserved, RES0.

### nDS, bit [17]

Disable Security not supported. Read-only and writes are ignored. Possible values are:

nDS	Meaning
0	The CPU interface logic supports disabling of security.
1	The CPU interface logic does not support disabling of security, and requires that security is not disabled.

### Bit [16]

Reserved, RES0.

### A3V, bit [15]

Affinity 3 Valid. Read-only and writes are ignored. Possible values are:

A3V	Meaning
0	The CPU interface logic does not support non-zero values of the Aff3 field in SGI generation System registers.
1	The CPU interface logic supports non-zero values of the Aff3 field in SGI generation System registers.



If EL3 is present, [ICC\\_CTLR](#).AV3 is an alias of ICC\_MCTLR.A3V

### SEIS, bit [14]

SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports generation of SEIs:

SEIS	Meaning
0	The CPU interface logic does not support generation of SEIs.
1	The CPU interface logic supports generation of SEIs.

If EL3 is present, [ICC\\_CTLR](#).SEIS is an alias of ICC\_MCTLR.SEIS

### IDbits, bits [13:11]

Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported:

IDbits	Meaning
000	16 bits.
001	24 bits.

All other values are reserved.

If EL3 is present, [ICC\\_CTLR](#).IDbits is an alias of ICC\_MCTLR.IDbits

### PRibits, bits [10:8]

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).

An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).

#### Note

This field always returns the number of priority bits implemented, regardless of the value of SCR.NS or the value of [GICD\\_CTLR](#).DS.

The division between group priority and subpriority is defined in the binary point registers [ICC\\_BPR0](#) and [ICC\\_BPR1](#).

This field determines the minimum value of ICC\_BPR0.

### Bit [7]

Reserved, RES0.

### PMHE, bit [6]

Priority Mask Hint Enable.

PMHE	Meaning
0	Disables use of the priority mask register as a hint for interrupt distribution.
1	Enables use of the priority mask register as a hint for interrupt distribution.

Software must write [ICC\\_PMR](#) to 0xFF before clearing this field to 0.

An implementation might choose to make this field RAO/WI.

If EL3 is present, [ICC\\_CTLR](#).PMHE is an alias of ICC\_MCTLR.PMHE.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**RM, bit [5]**

SBZ.

The equivalent bit in AArch64 is the Routing Modifier bit. This feature is not supported when EL3 is using AArch32.

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

**EOImode\_EL1NS, bit [4]**

EOI mode for interrupts handled at Non-secure EL1 and EL2. Controls whether a write to an End of Interrupt register also deactivates the interrupt:

EOImode_EL1NS	Meaning
0	<a href="#">ICC_EOIR0</a> and <a href="#">ICC_EOIR1</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">ICC_DIR</a> are UNPREDICTABLE.
1	<a href="#">ICC_EOIR0</a> and <a href="#">ICC_EOIR1</a> provide priority drop functionality only. <a href="#">ICC_DIR</a> provides interrupt deactivation functionality.

If EL3 is present, [ICC\\_CTLR](#)(NS).EOImode is an alias of ICC\_MCTLR.EOImode\_EL1NS.

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

**EOImode\_EL1S, bit [3]**

EOI mode for interrupts handled at Secure EL1. Controls whether a write to an End of Interrupt register also deactivates the interrupt:

EOImode_EL1S	Meaning
0	<a href="#">ICC_EOIR0</a> and <a href="#">ICC_EOIR1</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">ICC_DIR</a> are UNPREDICTABLE.
1	<a href="#">ICC_EOIR0</a> and <a href="#">ICC_EOIR1</a> provide priority drop functionality only. <a href="#">ICC_DIR</a> provides interrupt deactivation functionality.

If EL3 is present, [ICC\\_CTLR](#)(S).EOImode is an alias of ICC\_MCTLR.EOImode\_EL1S.

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

**EOImode\_EL3, bit [2]**

EOI mode for interrupts handled at EL3. Controls whether a write to an End of Interrupt register also deactivates the interrupt:

EOImode_EL3	Meaning
0	<a href="#">ICC_EOIR0</a> and <a href="#">ICC_EOIR1</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">ICC_DIR</a> are UNPREDICTABLE.
1	<a href="#">ICC_EOIR0</a> and <a href="#">ICC_EOIR1</a> provide priority drop functionality only. <a href="#">ICC_DIR</a> provides interrupt deactivation functionality.

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

**CBPR\_EL1NS, bit [1]**

Common Binary Point Register, EL1 Non-secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Non-secure interrupts at EL1 and EL2:

CBPR_EL1NS	Meaning
0	<a href="#">ICC_BPR0</a> determines the preemption group for Group 0 interrupts only. <a href="#">ICC_BPR1</a> determines the preemption group for Non-secure Group 1 interrupts.
1	<a href="#">ICC_BPR0</a> determines the preemption group for Group 0 interrupts and Non-secure Group 1 interrupts. Non-secure accesses to <a href="#">GICC_BPR</a> and <a href="#">ICC_BPR1</a> access the state of <a href="#">ICC_BPR0</a> .

If EL3 is present, [ICC\\_CTLR](#)(NS).CBPR is an alias of ICC\_MCTLR.CBPR\_EL1NS.

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

**CBPR\_EL1S, bit [0]**

Common Binary Point Register, EL1 Secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Secure interrupts in Secure non-Monitor modes:

CBPR_EL1S	Meaning
0	<a href="#">ICC_BPR0</a> determines the preemption group for Group 0 interrupts only. <a href="#">ICC_BPR1</a> determines the preemption group for Secure Group 1 interrupts.
1	<a href="#">ICC_BPR0</a> determines the preemption group for Group 0 interrupts and Secure Group 1 interrupts. Secure EL1 accesses, or EL3 accesses when not in Monitor mode, to <a href="#">ICC_BPR1</a> access the state of <a href="#">ICC_BPR0</a> .

If EL3 is present, [ICC\\_CTLR\(S\)](#).CBPR is an alias of ICC\_MCTLR.CBPR\_EL1S.

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

**Accessing the ICC\_MCTLR**

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, 6, <Rt>, c12, c12, 4	110	100	1100	1111	1100

**Accessibility**

The register is accessible as follows:

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

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

This register is only accessible when executing in Monitor mode.

**Traps and enables**

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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\\_MSRE](#).SRE==0, accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# ICC\_MGRPEN1, Interrupt Controller Monitor Interrupt Group 1 Enable register

The ICC\_MGRPEN1 characteristics are:

## Purpose

Controls whether Group 1 interrupts are enabled or not.

This register is part of:

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

## Configuration

This register is only accessible in Secure state.

AArch32 System register ICC\_MGRPEN1 can be mapped to AArch64 System register [ICC\\_IGRPEN1\\_EL3](#), but this is not architecturally mandated.

## Attributes

ICC\_MGRPEN1 is a 32-bit register.

## Field descriptions

The ICC\_MGRPEN1 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	EnableGrp1S	EnableGrp1NS

### Bits [31:2]

Reserved, RES0.

### EnableGrp1S, bit [1]

Enables Group 1 interrupts for the Secure state.

EnableGrp1S	Meaning
0	Secure Group 1 interrupts are disabled.
1	Secure Group 1 interrupts are enabled.

The Secure [ICC\\_IGRPEN1](#).Enable bit is a read/write alias of the ICC\_MGRPEN1.EnableGrp1S bit.

If the highest priority pending interrupt for that PE is a Group 1 interrupt using 1 of N model, then the interrupt will target another PE as a result of the Enable bit changing from 1 to 0.

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

### EnableGrp1NS, bit [0]

Enables Group 1 interrupts for the Non-secure state.

EnableGrp1NS	Meaning
0	Non-secure Group 1 interrupts are disabled.
1	Non-secure Group 1 interrupts are enabled.

The Non-secure [ICC\\_IGRPEN1](#).Enable bit is a read/write alias of the ICC\_MGRPEN1.EnableGrp1NS bit.

If the highest priority pending interrupt for that PE is a Group 1 interrupt using 1 of N model, then the interrupt will target another PE as a result of the Enable bit changing from 1 to 0.

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

## Accessing the ICC\_MGRPEN1

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, 6, <Rt>, c12, c12, 7	110	111	1100	1111	1100

## Accessibility

The register is accessible as follows:

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

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

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

This register is only accessible when executing in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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\\_MSRE](#).SRE==0, accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.



# ICC\_MSRE, Interrupt Controller Monitor System Register Enable register

The ICC\_MSRE characteristics are:

## Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL3.

This register is part of:

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

## Configuration

This register is only accessible in Secure state.

AArch32 System register ICC\_MSRE can be mapped to AArch64 System register [ICC\\_SRE\\_EL3](#), but this is not architecturally mandated.

## Attributes

ICC\_MSRE is a 32-bit register.

## Field descriptions

The ICC\_MSRE 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	Enable	DIB	DFB	SRE

### Bits [31:4]

Reserved, RES0.

### Enable, bit [3]

Enable. Enables lower Exception level access to [ICC\\_SRE](#) and [ICC\\_HSRE](#).

Enable	Meaning
0	Secure EL1 accesses to Secure <a href="#">ICC_SRE</a> trap to EL3. EL2 accesses to Non-secure <a href="#">ICC_SRE</a> and <a href="#">ICC_HSRE</a> trap to EL3. Non-secure EL1 accesses to <a href="#">ICC_SRE</a> trap to EL3, unless these accesses are trapped to EL2 as a result of ICC_MSRE.Enable == 0.
1	Secure EL1 accesses to Secure <a href="#">ICC_SRE</a> do not trap to EL3. EL2 accesses to Non-secure <a href="#">ICC_SRE</a> and <a href="#">ICC_HSRE</a> do not trap to EL3. Non-secure EL1 accesses to <a href="#">ICC_SRE</a> do not trap to EL3.

If ICC\_MSRE.SRE is RAO/WI, an implementation is permitted to make the Enable bit RAO/WI.

If ICC\_MSRE.SRE is 0, the Enable bit behaves as 1 for all purposes other than reading the value of the bit.

### DIB, bit [2]

Disable IRQ bypass.



DIB	Meaning
0	IRQ bypass enabled.
1	IRQ bypass disabled.

In systems that do not support IRQ bypass, 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 0.

### DFB, bit [1]

Disable FIQ bypass.

DFB	Meaning
0	FIQ bypass enabled.
1	FIQ bypass disabled.

In systems that do not support FIQ bypass, 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 0.

### SRE, bit [0]

System Register Enable.

SRE	Meaning
0	The memory-mapped interface must be used. Accesses at EL3 or below to any ICH_* System register, or any EL1, EL2, or EL3 ICC_* register other than <a href="#">ICC_SRE</a> , <a href="#">ICC_HSRE</a> , or ICC_MSRE, are UNDEFINED.
1	The System register interface to the ICH_* registers and the EL1, EL2, and EL3 ICC_* registers is enabled for EL3.

If software changes this bit from 1 to 0, the results are UNPREDICTABLE.

If an implementation supports only a System register interface to the GIC CPU interface, this bit is RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

## Accessing the ICC\_MSRE

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, 6, <Rt>, c12, c12, 5	110	101	1100	1111	1100

## Accessibility

The register is accessible as follows:

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

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

This register is always System register accessible.

The GIC architecture permits, but does not require, that registers can be shared between memory-mapped registers and the equivalent System registers. This means that if the memory-mapped registers have been accessed while  $ICC\_MSRE.SRE=0$ , then the System registers might be modified. Therefore, software must only rely on the reset values of the System registers if there has been no use of the GIC functionality while the memory-mapped registers are in use. Otherwise, the System register values must be treated as UNKNOWN.

This register is only accessible when executing in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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  $SCR\_EL3.NS=1 \ \&\& \ .E2H=0$  :

- If [HSTR\\_EL2.T12=1](#), Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and  $SCR\_EL3.NS=1 \ \&\& \ .E2H=1 \ \&\& \ HCR\_EL2.TGE=0$  :

- 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.

02/05/2017 15:43

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

# ICC\_PMR, Interrupt Controller Interrupt Priority Mask Register

The ICC\_PMR characteristics are:

## Purpose

Provides an interrupt priority filter. Only interrupts with a higher priority than the value in this register are signaled to the PE.

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\_PMR is architecturally mapped to AArch64 System register [ICC\\_PMR\\_EL1](#).

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that writes to this register are self-synchronising. This ensures that no interrupts below the written PMR value will be taken after a write to this register is architecturally executed. See Observability of the effects of accesses to the GIC registers, for more information.

## Attributes

ICC\_PMR is a 32-bit register.

## Field descriptions

The ICC\_PMR 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	Priority							

### Bits [31:8]

Reserved, RES0.

### Priority, bits [7:0]

The priority mask level for the CPU interface. If the priority of an interrupt is higher than the value indicated by this field, the interface signals the interrupt to the PE.

The possible priority field values are as follows:

Implemented priority bits	Possible priority field values	Number of priority levels
[7:0]	0x00-0xFF (0-255), all values	256
[7:1]	0x00-0xFE (0-254), even values only	128
[7:2]	0x00-0xFC (0-252), in steps of 4	64
[7:3]	0x00-0xF8 (0-248), in steps of 8	32
[7:4]	0x00-0xF0 (0-240), in steps of 16	16

Unimplemented priority bits are RAZ/WI.

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

## Accessing the ICC\_PMR

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>, c4, c6, 0	000	000	0100	1111	0110

When  $HCR.\{FMO, IMO\} \neq \{0, 0\}$ , execution of this encoding at Non-secure EL1 results in an access to [ICV\\_PMR](#).

## 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
x	1	0	1	-	<a href="#">ICV_PMR</a>	RW	RW
1	x	0	1	-	<a href="#">ICV_PMR</a>	RW	RW
0	0	0	1	-	RW	RW	RW

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

ICC\_PMR is only accessible at Non-secure EL1 when  $HCR.\{FMO, IMO\} = \{0, 0\}$ .

### Note

When  $HCR.\{FMO, IMO\} \neq \{0, 0\}$ , at Non-secure EL1, the instruction encoding used to access ICC\_PMR results in an access to [ICV\\_PMR](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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  $SCR\_EL3.NS=1$  :

- If [ICH\\_HCR](#).TC==1, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TC==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, and [SCR](#).FIQ==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, [SCR](#).FIQ==1, [HCR](#).IMO==0, and [HCR](#).FMO==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](#).FIQ==1, and [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](#).FIQ==1, and [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](#).FIQ==1, [SCR\\_EL3](#).IRQ==1, [HCR](#).IMO==0, and [HCR](#).FMO==0, Non-secure accesses to this register from EL1 are trapped to EL3.
- If [SCR\\_EL3](#).FIQ==1, [SCR\\_EL3](#).IRQ==1, [HCR\\_EL2](#).IMO==0, and [HCR\\_EL2](#).FMO==0, Non-secure accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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

# ICC\_RPR, Interrupt Controller Running Priority Register

The ICC\_RPR characteristics are:

## Purpose

Indicates the Running priority of the CPU interface.

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\_RPR performs the same function as AArch64 System register [ICC\\_RPR\\_EL1](#).

## Attributes

ICC\_RPR is a 32-bit register.

## Field descriptions

The ICC\_RPR 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	Priority									

### Bits [31:8]

Reserved, RES0.

### Priority, bits [7:0]

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

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

---

#### Note

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

---

## Accessing the ICC\_RPR

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>, c12, c11, 3	000	011	1100	1111	1011

When  $HCR.\{FMO, IMO\} \neq \{0, 0\}$ , execution of this encoding at Non-secure EL1 results in an access to [ICV\\_RPR](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	RO	n/a	RO
x	x	1	1	-	n/a	RO	RO
x	1	0	1	-	<a href="#">ICV_RPR</a>	RO	RO
1	x	0	1	-	<a href="#">ICV_RPR</a>	RO	RO
0	0	0	1	-	RO	RO	RO

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

ICC\_RPR is only accessible at Non-secure EL1 when  $HCR.\{FMO, IMO\} = \{0, 0\}$ .

### Note

When  $HCR.\{FMO, IMO\} \neq \{0, 0\}$ , at Non-secure EL1, the instruction encoding used to access ICC\_RPR results in an access to [ICV\\_RPR](#).

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

Software cannot determine the number of implemented priority bits from a read of 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<sup>®</sup> 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, read accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_HSRE](#).SRE==0, read accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, read accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and  $SCR\_EL3.NS=1 \ \&\& \ .E2H=0$  :

- If [HSTR\\_EL2](#).T12==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and  $SCR\_EL3.NS=1 \ \&\& \ .E2H=1 \ \&\& \ HCR\_EL2.TGE=0$  :

- If [HSTR\\_EL2](#).T12==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](#).T12==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When  $SCR\_EL3.NS=1$  :

- If [ICH\\_HCR](#).TC==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch32 :

- If [SCR](#).IRQ==1, and [SCR](#).FIQ==1, read 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, [SCR](#).FIQ==1, [HCR](#).IMO==0, and [HCR](#).FMO==0, Non-secure read 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 [SCR\\_EL3](#).IRQ==1, Secure read accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).FIQ==1, and [SCR\\_EL3](#).IRQ==1, read 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, [SCR\\_EL3](#).IRQ==1, [HCR](#).IMO==0, and [HCR](#).FMO==0, Non-secure read accesses to this register from EL1 are trapped to EL3.
- If [SCR\\_EL3](#).FIQ==1, [SCR\\_EL3](#).IRQ==1, [HCR\\_EL2](#).IMO==0, and [HCR\\_EL2](#).FMO==0, Non-secure read accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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



# ICC\_SGI0R, Interrupt Controller Software Generated Interrupt Group 0 Register

The ICC\_SGI0R characteristics are:

## Purpose

Generates Secure Group 0 SGIs.

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\_SGI0R performs the same function as AArch64 System register [ICC\\_SGI0R\\_EL1](#).

## Attributes

ICC\_SGI0R is a 64-bit register.

## Field descriptions

The ICC\_SGI0R 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	Aff3								0	0	0	0	0	0	0	0	IRM	Aff2									
0	0	0	0	INTID				Aff1								TargetList																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			

### Bits [63:56]

Reserved, RES0.

### Aff3, bits [55:48]

The affinity 3 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

### Bits [47:41]

Reserved, RES0.

### IRM, bit [40]

Interrupt Routing Mode. Determines how the generated interrupts are distributed to PEs. Possible values are:

IRM	Meaning
0	Interrupts routed to the PEs specified by Aff3.Aff2.Aff1.<target list>.
1	Interrupts routed to all PEs in the system, excluding "self".

**Aff2, bits [39:32]**

The affinity 2 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

**Bits [31:28]**

Reserved, RES0.

**INTID, bits [27:24]**

The INTID of the SGI.

**Aff1, bits [23:16]**

The affinity 1 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

**TargetList, bits [15:0]**

Target List. The set of PEs for which SGI interrupts will be generated. Each bit corresponds to the PE within a cluster with an Affinity 0 value equal to the bit number.

If a bit is 1 and the bit does not correspond to a valid target PE, the bit must be ignored by the Distributor. It is IMPLEMENTATION DEFINED whether, in such cases, a Distributor can signal a system error.

**Note**

This restricts a system to sending targeted SGIs to PEs with an affinity 0 number that is less than 16.

If SRE is set only for Secure EL3, software executing at EL3 might use the System register interface to generate SGIs. Therefore, the Distributor must always be able to receive and acknowledge Generate SGI packets received from CPU interface regardless of the ARE settings for a Security state. However, the Distributor might discard such packets.

If the IRM bit is 1, this field is RES0.

## Accessing the ICC\_SGI0R

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	coproc	CRm
p15, 2, <Rt>, <CRn>, c12, <opc2>	0010	1111	1100

## Accessibility

The register is accessible as follows:

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

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

This register allows software executing in a Secure state to generate Group 0 SGIs. It will also allow software executing in a Non-secure state to generate Group 0 SGIs, if permitted by the settings of [GICR\\_NSACR](#) in the Redistributor corresponding to the target PE.

When [GICD\\_CTLR.DS](#)==0, Non-secure writes do not generate an interrupt for a target PE if not permitted by the [GICR\\_NSACR](#) register associated with the target PE. For more information see Use of control registers for SGI forwarding.

---

#### Note

Accesses from Secure Monitor mode are treated as Secure regardless of the value of SCR.NS.

---

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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, write accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_HSRE.SRE](#)==0, write accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE.SRE](#)==0, write accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and [SCR\\_EL3.NS](#)==1 && [.E2H](#)==0 :

- If [HCR\\_EL2.FMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR\\_EL2.IMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HSTR\\_EL2.T12](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and [SCR\\_EL3.NS](#)==1 && [.E2H](#)==1 && [HCR\\_EL2.TGE](#)==0 :

- If [HCR\\_EL2.FMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR\\_EL2.IMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HSTR\\_EL2.T12](#)==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 [HCR.FMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR.IMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HSTR.T12](#)==1, Non-secure write accesses to this register from EL1 are trapped to Hyp mode.

When [SCR\\_EL3.NS](#)==1 :

- If [ICH\\_HCR.TC](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2.TC](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch32 :

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

When EL3 is implemented and is using AArch64 and [SCR\\_EL3.NS](#)==0 :

- If [SCR\\_EL3.FIQ](#)==1, and [SCR\\_EL3.IRQ](#)==1, Secure write accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3.FIQ](#)==1, and [SCR\\_EL3.IRQ](#)==1, write 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](#), [SCR\\_EL3.IRQ==1](#), [HCR.IMO==0](#), and [HCR.FMO==0](#), Non-secure write accesses to this register from EL1 are trapped to EL3.
- If [SCR\\_EL3.FIQ==1](#), [SCR\\_EL3.IRQ==1](#), [HCR\\_EL2.IMO==0](#), and [HCR\\_EL2.FMO==0](#), Non-secure write accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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

# ICC\_SGI1R, Interrupt Controller Software Generated Interrupt Group 1 Register

The ICC\_SGI1R characteristics are:

## Purpose

Generates Group 1 SGIs for the current Security state.

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\_SGI1R performs the same function as AArch64 System register [ICC\\_SGI1R\\_EL1](#).

Under certain conditions a write to ICC\_SGI1R can generate Group 0 interrupts, see Forwarding an SGI to a target PE.

## Attributes

ICC\_SGI1R is a 64-bit register.

## Field descriptions

The ICC\_SGI1R 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	Aff3								0	0	0	0	0	0	0	IRM	Aff2									
0	0	0	0	INTID				Aff1								TargetList																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

### Bits [63:56]

Reserved, RES0.

### Aff3, bits [55:48]

The affinity 3 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

### Bits [47:41]

Reserved, RES0.

### IRM, bit [40]

Interrupt Routing Mode. Determines how the generated interrupts are distributed to PEs. Possible values are:

IRM	Meaning
0	Interrupts routed to the PEs specified by Aff3.Aff2.Aff1.<target list>.
1	Interrupts routed to all PEs in the system, excluding "self".

**Aff2, bits [39:32]**

The affinity 2 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

**Bits [31:28]**

Reserved, RES0.

**INTID, bits [27:24]**

The INTID of the SGI.

**Aff1, bits [23:16]**

The affinity 1 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

**TargetList, bits [15:0]**

Target List. The set of PEs for which SGI interrupts will be generated. Each bit corresponds to the PE within a cluster with an Affinity 0 value equal to the bit number.

If a bit is 1 and the bit does not correspond to a valid target PE, the bit must be ignored by the Distributor. It is IMPLEMENTATION DEFINED whether, in such cases, a Distributor can signal a system error.

**Note**

This restricts a system to sending targeted SGIs to PEs with an affinity 0 number that is less than 16.

If SRE is set only for Secure EL3, software executing at EL3 might use the System register interface to generate SGIs. Therefore, the Distributor must always be able to receive and acknowledge Generate SGI packets received from CPU interface regardless of the ARE settings for a Security state. However, the Distributor might discard such packets.

If the IRM bit is 1, this field is RES0.

## Accessing the ICC\_SGI1R

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	coproc	CRm
p15, 0, <Rt>, <CRn>, c12, <opc2>	0000	1111	1100

## Accessibility

The register is accessible as follows:

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

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

## Note

Accesses from Secure Monitor mode are treated as Secure regardless of the value of SCR.NS.

## 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, write accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_HSRE](#).SRE==0, write accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, write accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HCR\\_EL2](#).FMO==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR\\_EL2](#).IMO==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HSTR\\_EL2](#).T12==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HCR\\_EL2](#).FMO==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR\\_EL2](#).IMO==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HSTR\\_EL2](#).T12==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 [HCR](#).FMO==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR](#).IMO==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HSTR](#).T12==1, Non-secure write accesses to this register from EL1 are trapped to Hyp mode.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR](#).TC==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch32 :

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

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).FIQ==1, and [SCR\\_EL3](#).IRQ==1, Secure write accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).FIQ==1, and [SCR\\_EL3](#).IRQ==1, write 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, [SCR\\_EL3](#).IRQ==1, [HCR](#).IMO==0, and [HCR](#).FMO==0, Non-secure write accesses to this register from EL1 are trapped to EL3.

- If [SCR\\_EL3.FIQ](#)=1, [SCR\\_EL3.IRQ](#)=1, [HCR\\_EL2.IMO](#)=0, and [HCR\\_EL2.FMO](#)=0, Non-secure write accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

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



# ICC\_SRE, Interrupt Controller System Register Enable register

The ICC\_SRE characteristics are:

## Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL0 and EL1.

This register is part of:

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

## Configuration

AArch32 System register ICC\_SRE (S) is architecturally mapped to AArch64 System register [ICC\\_SRE\\_EL1 \(S\)](#).

AArch32 System register ICC\_SRE (NS) is architecturally mapped to AArch64 System register [ICC\\_SRE\\_EL1 \(NS\)](#).

## Attributes

ICC\_SRE is a 32-bit register.

## Field descriptions

The ICC\_SRE 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	DIB	DFB	SRE

### Bits [31:3]

Reserved, RES0.

### DIB, bit [2]

Disable IRQ bypass.

DIB	Meaning
0	IRQ bypass enabled.
1	IRQ bypass disabled.

If EL3 is implemented and [GICD\\_CTLR.DS](#) == 0, this field is a read-only alias of [ICC\\_MSRE.DIB](#).

If EL3 is implemented and [GICD\\_CTLR.DS](#) == 1, and EL2 is not implemented, this field is a read-write alias of [ICC\\_MSRE.DIB](#).

If EL3 is not implemented or [GICD\\_CTLR.DS](#) == 1, and EL2 is implemented, this field is a read-only alias of [ICC\\_HSRE.DIB](#).

In systems that do not support IRQ bypass, this field is RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

### DFB, bit [1]

Disable FIQ bypass.

DFB	Meaning
0	FIQ bypass enabled.
1	FIQ bypass disabled.

If EL3 is implemented and [GICD\\_CTLR.DS](#) == 0, this field is a read-only alias of [ICC\\_MSRE.DFB](#).

If EL3 is implemented and [GICD\\_CTLR.DS](#) == 1, and EL2 is not implemented, this field is a read-write alias of [ICC\\_MSRE.DFB](#).

If EL3 is not implemented or [GICD\\_CTLR.DS](#) == 1, and EL2 is implemented, this field is a read-only alias of [ICC\\_HSRE.DFB](#).

In systems that do not support FIQ bypass, this field is RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

## SRE, bit [0]

System Register Enable.

SRE	Meaning
0	The memory-mapped interface must be used. Accesses at EL1 to any ICC_* System register other than ICC_SRE are UNDEFINED.
1	The System register interface for the current Security state is enabled.

If software changes this bit from 1 to 0 in the Secure instance of this register, the results are UNPREDICTABLE.

If an implementation supports only a System register interface to the GIC CPU interface, this bit is RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

## Accessing the ICC\_SRE

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, 5	000	101	1100	1111	1100

## Accessibility

The register is accessible as follows:

Configuration	Control		Accessibility				Instance
	TGE	NS	EL0	EL1	EL2	EL3	
EL3 not implemented	x	0	-	RW	n/a	-	ICC_SRE
EL3 not implemented	0	1	-	RW	RW	-	ICC_SRE
EL3 not implemented	1	1	-	n/a	RW	-	ICC_SRE
EL3 using AArch64	0	1	-	RW	RW	-	ICC_SRE_ns
EL3 using AArch64	1	1	-	n/a	RW	-	ICC_SRE_ns
EL3 using AArch32	0	1	-	RW	RW	RW	ICC_SRE_ns
EL3 using AArch32	1	1	-	n/a	RW	RW	ICC_SRE_ns
EL3 using AArch64	x	0	-	RW	n/a	-	ICC_SRE_s
EL3 using AArch32	x	0	-	-	-	RW	ICC_SRE_s

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

The GIC architecture permits, but does not require, that registers can be shared between memory-mapped registers and the equivalent System registers. This means that if the memory-mapped registers have been accessed while `ICC_SRE.SRE==0`, then the System registers might be modified. Therefore, software must only rely on the reset values of the System registers if there has been no use of the GIC functionality while the memory-mapped registers are in use. Otherwise, the System register values must be treated as UNKNOWN.

## 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_MSRE.Enable==0`, accesses to this register from EL1 and EL2 are trapped to EL3.

When EL2 is implemented and is using AArch64 and `SCR_EL3.NS==1 && .E2H==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 AArch64 and `SCR_EL3.NS==1 && .E2H==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 `ICC_HSRE.Enable==0`, Non-secure accesses to this register from EL1 are trapped to EL2.
- If `ICC_SRE_EL2.Enable==0`, Non-secure accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

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

# ICH\_AP0R<n>, Interrupt Controller Hyp Active Priorities Group 0 Registers, n = 0 - 3

The ICH\_AP0R<n> characteristics are:

## Purpose

Provides information about Group 0 active priorities for EL2.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch32 System register ICH\_AP0R<n> is architecturally mapped to AArch64 System register [ICH\\_AP0R<n>\\_EL2](#).

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

## Attributes

ICH\_AP0R<n> is a 32-bit register.

## Field descriptions

The ICH\_AP0R<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
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

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

Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are:

P<x>	Meaning
0	There is no Group 0 interrupt active at the priority corresponding to that bit.
1	There is a Group 0 interrupt active at the priority corresponding to that bit.

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

If 5 bits of priority are implemented (bits [7:3] of priority), then there are 32 priority levels, and the active state of these priority levels are held in ICH\_AP0R0 in the bits corresponding to Priority[7:3].

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

- The active state of priority levels 0 - 124 are held in ICH\_AP0R0 in the bits corresponding to 0:Priority[6:2].
- The active state of priority levels 128 - 252 are held in ICH\_AP0R1 in the bits corresponding to 1:Priority[6:2].

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

- The active state of priority levels 0 - 62 are held in ICH\_AP0R0 in the bits corresponding to 00:Priority[5:1].
- The active state of priority levels 64 - 126 are held in ICH\_AP0R1 in the bits corresponding to 01:Priority[5:1].
- The active state of priority levels 128 - 190 are held in ICH\_AP0R2 in the bits corresponding to 10:Priority[5:1].
- The active state of priority levels 192 - 254 are held in ICH\_AP0R3 in the bits corresponding to 11:Priority[5:1].

### Note

Having the bit corresponding to a priority set to 1 in both ICH\_AP0R<n> and [ICH\\_AP1R<n>](#) might result in UNPREDICTABLE behavior of the interrupt prioritization system for virtual interrupts.

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

## Accessing the ICH\_AP0R<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
p15, 4, <Rt>, c12, c8, <opc2>	100	0:n<1:0>	1100	1111	1000

- <opc2> is in the range 0 - 3.

## Accessibility

The register is accessible as follows:

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

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

ICH\_AP0R1 is only implemented in implementations that support 6 or more bits of priority. ICH\_AP0R2 and ICH\_AP0R3 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

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

- ICH\_AP0R<n>.
- [ICH\\_AP1R<n>](#).

## 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\\_HSRE](#).SRE==0, accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, Non-secure accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# ICH\_AP1R<n>, Interrupt Controller Hyp Active Priorities Group 1 Registers, n = 0 - 3

The ICH\_AP1R<n> characteristics are:

## Purpose

Provides information about Group 1 active priorities for EL2.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch32 System register ICH\_AP1R<n> is architecturally mapped to AArch64 System register [ICH\\_AP1R<n>\\_EL2](#).

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

## Attributes

ICH\_AP1R<n> is a 32-bit register.

## Field descriptions

The ICH\_AP1R<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
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

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

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

P<x>	Meaning
0	There is no Group 1 interrupt active at the priority corresponding to that bit.
1	There is a Group 1 interrupt active at the priority corresponding to that bit.

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

If 5 bits of priority are implemented (bits [7:3] of priority), then there are 32 priority levels, and the active state of these priority levels are held in ICH\_AP1R0 in the bits corresponding to Priority[7:3].

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

- The active state of priority levels 0 - 124 are held in ICH\_AP1R0 in the bits corresponding to 0:Priority[6:2].
- The active state of priority levels 128 - 252 are held in ICH\_AP1R1 in the bits corresponding to 1:Priority[6:2].

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

- The active state of priority levels 0 - 62 are held in ICH\_AP1R0 in the bits corresponding to 00:Priority[5:1].
- The active state of priority levels 64 - 126 are held in ICH\_AP1R1 in the bits corresponding to 01:Priority[5:1].
- The active state of priority levels 128 - 190 are held in ICH\_AP1R2 in the bits corresponding to 10:Priority[5:1].
- The active state of priority levels 192 - 254 are held in ICH\_AP1R3 in the bits corresponding to 11:Priority[5:1].

### Note

Having the bit corresponding to a priority set to 1 in both [ICH\\_AP0R<n>](#) and ICH\_AP1R<n> might result in UNPREDICTABLE behavior of the interrupt prioritization system for virtual interrupts.

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

## Accessing the ICH\_AP1R<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
p15, 4, <Rt>, c12, c9, <opc2>	100	0:n<1:0>	1100	1111	1001

- <opc2> is in the range 0 - 3.

## Accessibility

The register is accessible as follows:

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

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

ICH\_AP1R1 is only implemented in implementations that support 6 or more bits of priority. ICH\_AP1R2 and ICH\_AP1R3 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

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

- [ICH\\_AP0R<n>](#).
- ICH\_AP1R<n>.

## 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\\_HSRE](#).SRE==0, accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, Non-secure accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# ICH\_EISR, Interrupt Controller End of Interrupt Status Register

The ICH\_EISR characteristics are:

## Purpose

Indicates which List registers have outstanding EOI maintenance interrupts.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch32 System register ICH\_EISR is architecturally mapped to AArch64 System register [ICH\\_EISR\\_EL2](#).

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

## Attributes

ICH\_EISR is a 32-bit register.

## Field descriptions

The ICH\_EISR 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	Status<n>, bit [n], for n = 0 to 15															

### Bits [31:16]

Reserved, RES0.

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

EOI maintenance interrupt status bit for List register <n>:

Status<n>	Meaning
0	List register <n>, <a href="#">ICH_LR&lt;n&gt;</a> , does not have an EOI maintenance interrupt.
1	List register <n>, <a href="#">ICH_LR&lt;n&gt;</a> , has an EOI maintenance interrupt that has not been handled.

For any ICH\_LR<n>, the corresponding status bit is set to 1 if all of the following are true:

- [ICH\\_LRC<n>](#).State is 0b00.
- [ICH\\_LRC<n>](#).HW is 0.
- [ICH\\_LRC<n>](#).EOI (bit [9]) is 1, indicating that when the interrupt corresponding to that List register is deactivated, a maintenance interrupt is asserted.

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

## Accessing the ICH\_EISR

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, 4, <Rt>, c12, c11, 3	100	011	1100	1111	1011

## Accessibility

The register is accessible as follows:

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

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

- If [ICC\\_HSRE](#).SRE==0, read accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, Non-secure read accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

# ICH\_ELRSR, Interrupt Controller Empty List Register Status Register

The ICH\_ELRSR characteristics are:

## Purpose

Indicates which List registers contain valid interrupts.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch32 System register ICH\_ELRSR is architecturally mapped to AArch64 System register [ICH\\_ELRSR\\_EL2](#).

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

## Attributes

ICH\_ELRSR is a 32-bit register.

## Field descriptions

The ICH\_ELRSR 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	Status<n>, bit [n], for n = 0 to 15															

### Bits [31:16]

Reserved, RES0.

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

Status bit for List register <n>, [ICH\\_LR<n>](#):

Status<n>	Meaning
0	List register <a href="#">ICH_LR&lt;n&gt;</a> , if implemented, contains a valid interrupt. Using this List register can result in overwriting a valid interrupt.
1	List register <a href="#">ICH_LR&lt;n&gt;</a> does not contain a valid interrupt. The List register is empty and can be used without overwriting a valid interrupt or losing an EOI maintenance interrupt.

For any List register <n>, the corresponding status bit is set to 1 if [ICH\\_LRC<n>](#).State is 0b00 and either [ICH\\_LRC<n>](#).HW is 1 or [ICH\\_LRC<n>](#).EOI (bit [9]) is 0.

## Accessing the ICH\_ELRSR

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, 4, <Rt>, c12, c11, 5	100	101	1100	1111	1011

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	-	n/a	-
0	1	-	-	RO	RO
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<sup>®</sup> 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\\_HSRE](#).SRE==0, read accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, Non-secure read accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

# ICH\_HCR, Interrupt Controller Hyp Control Register

The ICH\_HCR characteristics are:

## Purpose

Controls the environment for VMs.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch32 System register ICH\_HCR is architecturally mapped to AArch64 System register [ICH\\_HCR\\_EL2](#).

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

## Attributes

ICH\_HCR is a 32-bit register.

## Field descriptions

The ICH\_HCR 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
EOIcount	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TDIR	TSEI	TALL1	TALL0	TC	0	0	VGrp1DIE	VGrp1EIE	VGrp0DIE	VGrp0EIE	NPIEL	RENPIE	UIE	En

### EOIcount, bits [31:27]

This field is incremented whenever a successful write to a virtual EOIR or DIR register would have resulted in a virtual interrupt deactivation. That is either:

- A virtual write to EOIR with a valid interrupt identifier that is not in the LPI range (i.e. < 8192) when EOI mode is zero and no List Register was found.
- A virtual write to DIR with a valid interrupt identifier that is not in the LPI range (i.e. < 8192) when EOI mode is one and no List Register was found.

This allows software to manage more active interrupts than there are implemented List Registers.

It is CONSTRAINED UNPREDICTABLE whether a virtual write to EOIR that does not clear a bit in the Active Priorities registers ([ICH\\_APOR<n>](#)/[ICH\\_APIR<n>](#)) increments EOIcount. Permitted behaviors are:

- Increment EOIcount.
- Leave EOIcount unchanged.

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

### Bits [26:15]

Reserved, RES0.

### TDIR, bit [14]

Trap Non-secure EL1 writes to [ICC\\_DIR](#) and [ICV\\_DIR](#).

<b>TDIR</b>	<b>Meaning</b>
0	Non-secure EL1 writes of <a href="#">ICC_DIR</a> and <a href="#">ICV_DIR</a> are not trapped to EL2, unless trapped by other mechanisms.
1	Non-secure EL1 writes of <a href="#">ICC_DIR</a> and <a href="#">ICV_DIR</a> are trapped to EL2.

Support for this bit is OPTIONAL, with support indicated by [ICH\\_VTR](#).

If the implementation does not support this trap, this bit is RES0.

ARM deprecates not including this trap bit.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### TSEI, bit [13]

Trap all locally generated SEIs. This bit allows the hypervisor to intercept locally generated SEIs that would otherwise be taken at Non-secure EL1.

<b>TSEI</b>	<b>Meaning</b>
0	Locally generated SEIs do not cause a trap to EL2.
1	Locally generated SEIs trap to EL2.

If [ICH\\_VTR](#).SEIS is 0, 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.

#### TALL1, bit [12]

Trap all Non-secure EL1 accesses to ICC\_\* and ICV\_\* System registers for Group 1 interrupts to EL2.

<b>TALL1</b>	<b>Meaning</b>
0	Non-Secure EL1 accesses to ICC_* and ICV_* registers for Group 1 interrupts proceed as normal.
1	Non-secure EL1 accesses to ICC_* and ICV_* registers for Group 1 interrupts trap to EL2.

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

#### TALL0, bit [11]

Trap all Non-secure EL1 accesses to ICC\_\* and ICV\_\* System registers for Group 0 interrupts to EL2.

<b>TALL0</b>	<b>Meaning</b>
0	Non-Secure EL1 accesses to ICC_* and ICV_* registers for Group 0 interrupts proceed as normal.
1	Non-secure EL1 accesses to ICC_* and ICV_* registers for Group 0 interrupts trap to EL2.

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

#### TC, bit [10]

Trap all Non-secure EL1 accesses to System registers that are common to Group 0 and Group 1 to EL2.

<b>TC</b>	<b>Meaning</b>
0	Non-secure EL1 accesses to common registers proceed as normal.
1	Non-secure EL1 accesses to common registers trap to EL2.

This affects accesses to [ICC\\_SGI0R](#), [ICC\\_SGI1R](#), [ICC\\_ASGI1R](#), [ICC\\_CTLR](#), [ICC\\_DIR](#), [ICC\\_PMR](#), [ICC\\_RPR](#), [ICV\\_SGI0R](#), [ICV\\_SGI1R](#), [ICV\\_ASGI1R](#), [ICV\\_CTLR](#), [ICV\\_DIR](#), [ICV\\_PMR](#), and [ICV\\_RPR](#).

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

**Bits [9:8]**

Reserved, RES0.

**VGrp1DIE, bit [7]**

VM Group 1 Disabled Interrupt Enable. Enables the signaling of a maintenance interrupt while signaling of Group 1 interrupts from the virtual CPU interface to the connected vPE is disabled:

VGrp1DIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled when <a href="#">ICH_VMCR.VENG1</a> is 0.

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

**VGrp1EIE, bit [6]**

VM Group 1 Enabled Interrupt Enable. Enables the signaling of a maintenance interrupt while signaling of Group 1 interrupts from the virtual CPU interface to the connected vPE is enabled:

VGrp1EIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled when <a href="#">ICH_VMCR.VENG1</a> is 1.

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

**VGrp0DIE, bit [5]**

VM Group 0 Disabled Interrupt Enable. Enables the signaling of a maintenance interrupt while signaling of Group 0 interrupts from the virtual CPU interface to the connected vPE is disabled:

VGrp0DIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled when <a href="#">ICH_VMCR.VENG0</a> is 0.

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

**VGrp0EIE, bit [4]**

VM Group 0 Enabled Interrupt Enable. Enables the signaling of a maintenance interrupt while signaling of Group 0 interrupts from the virtual CPU interface to the connected vPE is enabled:

VGrp0EIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled when <a href="#">ICH_VMCR.VENG0</a> is 1.

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

**NPIE, bit [3]**

No Pending Interrupt Enable. Enables the signaling of a maintenance interrupt while no pending interrupts are present in the List registers:

NPIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled while the List registers contain no interrupts in the pending state.

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

**LRENPIE, bit [2]**

List Register Entry Not Present Interrupt Enable. Enables the signaling of a maintenance interrupt while the virtual CPU interface does not have a corresponding valid List register entry for an EOI request:



<b>LRENPIE</b>	<b>Meaning</b>
0	Maintenance interrupt disabled.
1	Maintenance interrupt is asserted while the EOICount field is not 0.

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

### UIE, bit [1]

Underflow Interrupt Enable. Enables the signaling of a maintenance interrupt when the List registers are empty, or hold only one valid entry:

<b>UIE</b>	<b>Meaning</b>
0	Maintenance interrupt disabled.
1	Maintenance interrupt is asserted if none, or only one, of the List register entries is marked as a valid interrupt.

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

### En, bit [0]

Enable. Global enable bit for the virtual CPU interface:

<b>En</b>	<b>Meaning</b>
0	Virtual CPU interface operation disabled.
1	Virtual CPU interface operation enabled.

When this field is set to 0:

- The virtual CPU interface does not signal any maintenance interrupts.
- The virtual CPU interface does not signal any virtual interrupts.
- A read of [ICV\\_IAR0](#), [ICV\\_IAR1](#), [GICV\\_IAR](#) or [GICV\\_AIAR](#) returns a spurious interrupt ID.

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

## Accessing the ICH\_HCR

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:

<b>&lt;syntax&gt;</b>	<b>opc1</b>	<b>opc2</b>	<b>CRn</b>	<b>coproc</b>	<b>CRm</b>
p15, 4, <Rt>, c12, c11, 0	100	000	1100	1111	1011

## Accessibility

The register is accessible as follows:

<b>Control</b>		<b>Accessibility</b>			
<b>TGE</b>	<b>NS</b>	<b>EL0</b>	<b>EL1</b>	<b>EL2</b>	<b>EL3</b>
x	0	-	-	n/a	-
0	1	-	-	RW	RW
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 [ICC\\_HSRE](#).SRE==0, accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, Non-secure accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# ICH\_LR<n>, Interrupt Controller List Registers, n = 0 - 15

The ICH\_LR<n> characteristics are:

## Purpose

Provides interrupt context information for the virtual CPU interface.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch32 System register ICH\_LR<n> is architecturally mapped to AArch64 System register [ICH\\_LR<n>\\_EL2\[31:0\]](#).

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

## Attributes

ICH\_LR<n> is a 32-bit register.

## Field descriptions

The ICH\_LR<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
																vINTID															

### vINTID, bits [31:0]

Virtual INTID of the interrupt.

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

- ICH\_LR<n>.State == 01.
- ICH\_LR<n>.State == 10.
- ICH\_LR<n>.State == 11.

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

---

### Note

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

---

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

## Accessing the ICH\_LR<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
p15, 4, <Rt>, c12, <CRm>, <opc2>	100	n<2:0>	1100	1111	110:n<3>

- <opc2> is in the range 0 - 7.
- <CRm> is in the range c12 - c13.

## Accessibility

The register is accessible as follows:

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

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

ICH\_LR<n> and [ICH\\_LRC<n>](#) can be updated independently.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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\\_HSRE](#).SRE==0, accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, Non-secure accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

# ICH\_LRC<n>, Interrupt Controller List Registers, n = 0 - 15

The ICH\_LRC<n> characteristics are:

## Purpose

Provides interrupt context information for the virtual CPU interface.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch32 System register ICH\_LRC<n> is architecturally mapped to AArch64 System register [ICH\\_LR<n>\\_EL2\[63:32\]](#).

## Attributes

ICH\_LRC<n> is a 32-bit register.

## Field descriptions

The ICH\_LRC<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
State	HW	Group	0	0	0	0					Priority					0	0	0	0	0	0										pINTID

### State, bits [31:30]

The state of the interrupt:

State	Meaning
00	Inactive
01	Pending
10	Active
11	Pending and active.

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

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

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

### HW, bit [29]

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

HW	Meaning
0	The interrupt is triggered entirely by software. No notification is sent to the Distributor when the virtual interrupt is deactivated.
1	The interrupt maps directly to a hardware interrupt. A deactivate interrupt request is sent to the Distributor when the virtual interrupt is deactivated, using the pINTID field from this register to indicate the physical INTID. If <a href="#">ICH_VMCR.VEOIM</a> is 0, this request corresponds to a write to <a href="#">ICC_EOIR0</a> or <a href="#">ICC_EOIR1</a> . Otherwise, it corresponds to a write to <a href="#">ICC_DIR</a> .

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

## Group, bit [28]

Indicates the group for this virtual interrupt.

Group	Meaning
0	This is a Group 0 virtual interrupt. <a href="#">ICH_VMCR.VFIQEn</a> determines whether it is signaled as a virtual IRQ or as a virtual FIQ, and <a href="#">ICH_VMCR.VENG0</a> enables signaling of this interrupt to the virtual machine.
1	This is a Group 1 virtual interrupt, signaled as a virtual IRQ. <a href="#">ICH_VMCR.VENG1</a> enables the signaling of this interrupt to the virtual machine. If <a href="#">ICH_VMCR.VCBPR</a> is 0, then <a href="#">ICC_BPR1</a> determines if a pending Group 1 interrupt has sufficient priority to preempt current execution. Otherwise, <a href="#">ICH_LR&lt;n&gt;</a> determines preemption.

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

## Bits [27:24]

Reserved, RES0.

## Priority, bits [23:16]

The priority of this interrupt.

It is IMPLEMENTATION DEFINED how many bits of priority are implemented, though at least five bits must be implemented. Unimplemented bits are RES0 and start from bit [16] up to bit [18]. The number of implemented bits can be discovered from [ICH\\_VTR.PRIBits](#).

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

## Bits [15:10]

Reserved, RES0.

## pINTID, bits [9:0]

Physical INTID, for hardware interrupts.

When the HW bit is 0 (there is no corresponding physical interrupt), this field has the following meaning:

- Bit [9] : EOI. If this bit is 1, then when the interrupt identified by vINTID is deactivated, an EOI maintenance interrupt is asserted.
- Bits [8:0] : Reserved, RES0.

When the HW bit is 1 (there is a corresponding physical interrupt):

- This field indicates the physical INTID. This field is only required to implement enough bits to hold a valid value for the implemented INTID size. Any unused higher order bits are RES0.
- If the value of pINTID is 0-15 or 1020-1023, behavior is UNPREDICTABLE. If the value of pINTID is 16-31, this field applies to the PPI associated with this same physical PE ID as the virtual CPU interface requesting the deactivation.

A hardware physical identifier is only required in List Registers for interrupts that require deactivation. This means only 10 bits of Physical INTID are required, regardless of the number specified by [ICC\\_CTLR.IDbits](#).

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

## Accessing the ICH\_LRC<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
p15, 4, <Rt>, c12, <CRm>, <opc2>	100	n<2:0>	1100	1111	111:n<3>

- <opc2> is in the range 0 - 7.
- <CRm> is in the range c14 - c15.

## Accessibility

The register is accessible as follows:

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

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

[ICH\\_LR<n>](#) and ICH\_LRC<n> can be updated independently.

## 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\\_HSRE](#).SRE==0, accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, Non-secure accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

# ICH\_MISR, Interrupt Controller Maintenance Interrupt State Register

The ICH\_MISR characteristics are:

## Purpose

Indicates which maintenance interrupts are asserted.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch32 System register ICH\_MISR is architecturally mapped to AArch64 System register [ICH\\_MISR\\_EL2](#).

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

## Attributes

ICH\_MISR is a 32-bit register.

## Field descriptions

The ICH\_MISR 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	VGrp1D	VGrp1E	VGrp0D	VGrp0E	NPL	REN	P	U	EOI

### Bits [31:8]

Reserved, RES0.

### VGrp1D, bit [7]

vPE Group 1 Disabled.

VGrp1D	Meaning
0	vPE Group 1 Disabled maintenance interrupt not asserted.
1	vPE Group 1 Disabled maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR](#).VGrp1DIE==1 and [ICH\\_VMCR](#).VMGrp1En==0.

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

### VGrp1E, bit [6]

vPE Group 1 Enabled.

VGrp1E	Meaning
0	vPE Group 1 Enabled maintenance interrupt not asserted.
1	vPE Group 1 Enabled maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR](#).VGrp1EIE==1 and [ICH\\_VMCR](#).VMGrp1En==1.



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

#### VGrp0D, bit [5]

vPE Group 0 Disabled.

VGrp0D	Meaning
0	vPE Group 0 Disabled maintenance interrupt not asserted.
1	vPE Group 0 Disabled maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR.VGrp0DIE](#)==1 and [ICH\\_VMCR.VMGrp0En](#)==0.

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

#### VGrp0E, bit [4]

vPE Group 0 Enabled.

VGrp0E	Meaning
0	vPE Group 0 Enabled maintenance interrupt not asserted.
1	vPE Group 0 Enabled maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR.VGrp0EIE](#)==1 and [ICH\\_VMCR.VMGrp0En](#)==1.

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

#### NP, bit [3]

No Pending.

NP	Meaning
0	No Pending maintenance interrupt not asserted.
1	No Pending maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR.NPIE](#)==1 and no List register is in pending state.

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

#### LREN, bit [2]

List Register Entry Not Present.

LREN	Meaning
0	List Register Entry Not Present maintenance interrupt not asserted.
1	List Register Entry Not Present maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR.LRENPIE](#)==1 and [ICH\\_HCR.EOIcount](#) is non-zero.

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

#### U, bit [1]

Underflow.

U	Meaning
0	Underflow maintenance interrupt not asserted.
1	Underflow maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR.UIE](#)==1 and zero or one of the List register entries are marked as a valid interrupt, that is, if the corresponding [ICH\\_LRC<n>.State](#) bits do not equal 0x0.

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

## EOI, bit [0]

End Of Interrupt.

EOI	Meaning
0	End Of Interrupt maintenance interrupt not asserted.
1	End Of Interrupt maintenance interrupt asserted.

This maintenance interrupt is asserted when at least one bit in [ICH\\_EISR](#) is 1.

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

## Accessing the ICH\_MISR

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, 4, <Rt>, c12, c11, 2	100	010	1100	1111	1011

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	-	n/a	-
0	1	-	-	RO	RO
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<sup>®</sup> 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\\_HSRE](#).SRE==0, read accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, Non-secure read accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.



# ICH\_VMCR, Interrupt Controller Virtual Machine Control Register

The ICH\_VMCR characteristics are:

## Purpose

Enables the hypervisor to save and restore the virtual machine view of the GIC state.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch32 System register ICH\_VMCR is architecturally mapped to AArch64 System register [ICH\\_VMCR\\_EL2](#).

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

## Attributes

ICH\_VMCR is a 32-bit register.

## Field descriptions

The ICH\_VMCR 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
VPMR								VBPR0	VBPR1	0	0	0	0	0	0	0	0	0	0	0	0	VEOIM	0	0	0	0	VCBPR	VFIQEn	VAckCtl	VENG1	VENG0

### VPMR, bits [31:24]

Virtual Priority Mask. The priority mask level for the virtual CPU interface. If the priority of a pending virtual interrupt is higher than the value indicated by this field, the interface signals the virtual interrupt to the PE.

This field is an alias of [ICV\\_PMR](#).Priority.

### VBPR0, bits [23:21]

Virtual Binary Point Register, Group 0. Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 0 interrupt preemption, and also determines Group 1 interrupt preemption if  $ICH\_VMCR.VCBPR == 1$ .

This field is an alias of [ICV\\_BPR0](#).BinaryPoint.

### VBPR1, bits [20:18]

Virtual Binary Point Register, Group 1. Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption if  $ICH\_VMCR.VCBPR == 0$ .

This field is an alias of [ICV\\_BPR1](#).BinaryPoint.

### Bits [17:10]

Reserved, RES0.

## VEOIM, bit [9]

Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt:

VEOIM	Meaning
0	<a href="#">ICV_EOIR0</a> and <a href="#">ICV_EOIR1</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">ICV_DIR</a> are UNPREDICTABLE.
1	<a href="#">ICV_EOIR0</a> and <a href="#">ICV_EOIR1</a> provide priority drop functionality only. <a href="#">ICV_DIR</a> provides interrupt deactivation functionality.

This bit is an alias of [ICV\\_CTLR](#).EOImode.

## Bits [8:5]

Reserved, RES0.

## VCBPR, bit [4]

Virtual Common Binary Point Register. Possible values of this bit are:

VCBPR	Meaning
0	<a href="#">ICV_BPR0</a> determines the preemption group for virtual Group 0 interrupts only. <a href="#">ICV_BPR1</a> determines the preemption group for virtual Group 1 interrupts.
1	<a href="#">ICV_BPR0</a> determines the preemption group for both virtual Group 0 and virtual Group 1 interrupts. Reads of <a href="#">ICV_BPR1</a> return <a href="#">ICV_BPR0</a> plus one, saturated to 0b111. Writes to <a href="#">ICV_BPR1</a> are ignored.

This field is an alias of [ICV\\_CTLR](#).CBPR.

## VFIQEn, bit [3]

Virtual FIQ enable. Possible values of this bit are:

VFIQEn	Meaning
0	Group 0 virtual interrupts are presented as virtual IRQs.
1	Group 0 virtual interrupts are presented as virtual FIQs.

This bit is an alias of [GICV\\_CTLR](#).FIQEn.

In implementations where the Non-secure copy of [ICC\\_SRE](#).SRE is always 1, this bit is RES1.

## VAckCtl, bit [2]

Virtual AckCtl. Possible values of this bit are:

VAckCtl	Meaning
0	If the highest priority pending interrupt is Group 1, a read of <a href="#">GICV_IAR</a> or <a href="#">GICV_HPPIR</a> returns an INTID of 1022.
1	If the highest priority pending interrupt is Group 1, a read of <a href="#">GICV_IAR</a> or <a href="#">GICV_HPPIR</a> returns the INTID of the corresponding interrupt.

This bit is an alias of [GICV\\_CTLR](#).AckCtl.

This field is supported for backwards compatibility with GICv2. ARM deprecates the use of this field.

In implementations where the Non-secure copy of [ICC\\_SRE](#).SRE is always 1, this bit is RES0.

## VENG1, bit [1]

Virtual Group 1 interrupt enable. Possible values of this bit are:

VENG1	Meaning
0	Virtual Group 1 interrupts are disabled.
1	Virtual Group 1 interrupts are enabled.

This bit is an alias of [ICV\\_IGRPEN1](#).Enable.

### VENG0, bit [0]

Virtual Group 0 interrupt enable. Possible values of this bit are:

VENG0	Meaning
0	Virtual Group 0 interrupts are disabled.
1	Virtual Group 0 interrupts are enabled.

This bit is an alias of [ICV\\_IGRPEN0](#).Enable.

## Accessing the ICH\_VMCR

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>, c12, c11, 7	100	111	1100	1111	1011

## Accessibility

The register is accessible as follows:

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

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

When EL2 is using System register access, EL1 using either System register or memory-mapped access must be supported.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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\\_HSRE](#).SRE==0, accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, Non-secure accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# ICH\_VTR, Interrupt Controller VGIC Type Register

The ICH\_VTR characteristics are:

## Purpose

Reports supported GIC virtualisartion features.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch32 System register ICH\_VTR is architecturally mapped to AArch64 System register [ICH\\_VTR\\_EL2](#).

If EL2 is not implemented, all bits in this register are RES0 from EL3, except for nV4, which is RES1 from EL3.

## Attributes

ICH\_VTR is a 32-bit register.

## Field descriptions

The ICH\_VTR 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
<a href="#">PRIBits</a>	<a href="#">PREbits</a>	<a href="#">IDbits</a>	<a href="#">SEIS</a>	<a href="#">A3V</a>	<a href="#">nV4</a>	<a href="#">TDS</a>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<a href="#">ListRegs</a>					

### PRIBits, bits [31:29]

Priority bits. The number of virtual priority bits implemented, minus one.

An implementation must implement at least 32 levels of virtual priority (5 priority bits).

This field is an alias of [ICV\\_CTLR](#).PRIBits.

### PREbits, bits [28:26]

The number of virtual preemption bits implemented, minus one.

An implementation must implement at least 32 levels of virtual preemption priority (5 preemption bits).

The value of this field must be less than or equal to the value of ICH\_VTR.PRIBits.

### IDbits, bits [25:23]

The number of virtual interrupt identifier bits supported:

IDbits	Meaning
000	16 bits.
001	24 bits.

All other values are reserved.

This field is an alias of [ICV\\_CTLR](#).IDbits.



### SEIS, bit [22]

SEI Support. Indicates whether the virtual CPU interface supports generation of SEIs:

SEIS	Meaning
0	The virtual CPU interface logic does not support generation of SEIs.
1	The virtual CPU interface logic supports generation of SEIs.

This bit is an alias of [ICV\\_CTLR](#).SEIS.

### A3V, bit [21]

Affinity 3 Valid. Possible values are:

A3V	Meaning
0	The virtual CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers.
1	The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

This bit is an alias of [ICV\\_CTLR](#).A3V.

### nV4, bit [20]

Direct injection of virtual interrupts not supported. Possible values are:

nV4	Meaning
0	The CPU interface logic supports direct injection of virtual interrupts.
1	The CPU interface logic does not support direct injection of virtual interrupts.

In GICv3 this bit is RES1.

### TDS, bit [19]

Separate trapping of Non-secure EL1 writes to [ICV\\_DIR](#) supported.

TDS	Meaning
0	Implementation does not support <a href="#">ICH_HCR</a> .TDIR.
1	Implementation supports <a href="#">ICH_HCR</a> .TDIR.

### Bits [18:5]

Reserved, RES0.

### ListRegs, bits [4:0]

The number of implemented List registers, minus one. For example, a value of 0b01111 indicates that the maximum of 16 List registers are implemented.

## Accessing the ICH\_VTR

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, 4, <Rt>, c12, c11, 1	100	001	1100	1111	1011

## Accessibility

The register is accessible as follows:

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

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

- If [ICC\\_HSRE](#).SRE==0, read accesses to this register from EL2 are UNDEFINED.
- If [ICC\\_MSRE](#).SRE==0, Non-secure read accesses to this register from EL3 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# ICV\_AP0R<n>, Interrupt Controller Virtual Active Priorities Group 0 Registers, n = 0 - 3

The ICV\_AP0R<n> characteristics are:

## Purpose

Provides information about virtual Group 0 active priorities.

This register is part of:

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

## Configuration

AArch32 System register ICV\_AP0R<n> is architecturally mapped to AArch64 System register [ICV\\_AP0R<n>\\_EL1](#).

## Attributes

ICV\_AP0R<n> is a 32-bit register.

## Field descriptions

The ICV\_AP0R<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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

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

The contents of these registers are IMPLEMENTATION DEFINED with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

## Accessing the ICV\_AP0R<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
p15, 0, <Rt>, c12, c8, <opc2>	000	1:n<1:0>	1100	1111	1000

- <opc2> is in the range 4 - 7.

When HCR.FMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_AP0R<n>](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_AP0R&lt;n&gt;</a>	n/a	<a href="#">ICC_AP0R&lt;n&gt;</a>
x	x	1	1	-	n/a	<a href="#">ICC_AP0R&lt;n&gt;</a>	<a href="#">ICC_AP0R&lt;n&gt;</a>
0	x	0	1	-	<a href="#">ICC_AP0R&lt;n&gt;</a>	<a href="#">ICC_AP0R&lt;n&gt;</a>	<a href="#">ICC_AP0R&lt;n&gt;</a>
1	x	0	1	-	RW	<a href="#">ICC_AP0R&lt;n&gt;</a>	<a href="#">ICC_AP0R&lt;n&gt;</a>

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

The ICV\_AP0R<n> registers are 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\_AP0R<n> results in an access to [ICC\\_AP0R<n>](#).

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 0 active priorities) might result in UNPREDICTABLE behavior of the virtual interrupt prioritization system, causing:

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

ICV\_AP0R1 is only implemented in implementations that support 6 or more bits of priority. ICV\_AP0R2 and ICV\_AP0R3 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

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

- ICV\_AP0R<n>.
- [ICV\\_APIR<n>](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.



# ICV\_AP1R<n>, Interrupt Controller Virtual Active Priorities Group 1 Registers, n = 0 - 3

The ICV\_AP1R<n> characteristics are:

## Purpose

Provides information about virtual Group 1 active priorities.

This register is part of:

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

## Configuration

AArch32 System register ICV\_AP1R<n> is architecturally mapped to AArch64 System register [ICV\\_AP1R<n>\\_EL1](#).

## Attributes

ICV\_AP1R<n> is a 32-bit register.

## Field descriptions

The ICV\_AP1R<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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

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

The contents of these registers are IMPLEMENTATION DEFINED with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

## Accessing the ICV\_AP1R<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
p15, 0, <Rt>, c12, c9, <opc2>	000	0:n<1:0>	1100	1111	1001

- <opc2> is in the range 0 - 3.

When HCR.IMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_AP1R<n>](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_AP1R&lt;n&gt;</a>	n/a	<a href="#">ICC_AP1R&lt;n&gt;</a>
x	x	1	1	-	n/a	<a href="#">ICC_AP1R&lt;n&gt;</a>	<a href="#">ICC_AP1R&lt;n&gt;</a>
x	0	0	1	-	<a href="#">ICC_AP1R&lt;n&gt;</a>	<a href="#">ICC_AP1R&lt;n&gt;</a>	<a href="#">ICC_AP1R&lt;n&gt;</a>
x	1	0	1	-	RW	<a href="#">ICC_AP1R&lt;n&gt;</a>	<a href="#">ICC_AP1R&lt;n&gt;</a>

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

The ICV\_AP1R<n> registers are only accessible at Non-secure EL1 when HCR.IMO == 1.

### Note

When HCR.IMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_AP1R<n> results in an access to [ICC\\_AP1R<n>](#).

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 1 active priorities) might result in UNPREDICTABLE behavior of the virtual interrupt prioritization system, causing:

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

ICV\_AP1R1 is only implemented in implementations that support 6 or more bits of priority. ICV\_AP1R2 and ICV\_AP1R3 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

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

- [ICV\\_AP0R<n>](#).
- ICV\_AP1R<n>.

## 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 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.





# ICV\_BPR0, Interrupt Controller Virtual Binary Point Register 0

The ICV\_BPR0 characteristics are:

## Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines virtual Group 0 interrupt preemption.

This register is part of:

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

## Configuration

AArch32 System register ICV\_BPR0 is architecturally mapped to AArch64 System register [ICV\\_BPR0\\_EL1](#).

## Attributes

ICV\_BPR0 is a 32-bit register.

## Field descriptions

The ICV\_BPR0 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	BinaryPoint

### Bits [31:3]

Reserved, RES0.

### BinaryPoint, bits [2:0]

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. This is done as follows:

Binary point value	Group priority field	Subpriority field	Field with binary point
0	[7:1]	[0]	ggggggg.s
1	[7:2]	[1:0]	gggggg.ss
2	[7:3]	[2:0]	ggggg.sss
3	[7:4]	[3:0]	gggg.ssss
4	[7:5]	[4:0]	ggg.sssss
5	[7:6]	[5:0]	gg.ssssss
6	[7]	[6:0]	g.sssssss
7	No preemption	[7:0]	.ssssssss

## Accessing the ICV\_BPR0

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, c8, 3	000	011	1100	1111	1000

When HCR.FMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_BPR0](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_BPR0</a>	n/a	<a href="#">ICC_BPR0</a>
x	x	1	1	-	n/a	<a href="#">ICC_BPR0</a>	<a href="#">ICC_BPR0</a>
0	x	0	1	-	<a href="#">ICC_BPR0</a>	<a href="#">ICC_BPR0</a>	<a href="#">ICC_BPR0</a>
1	x	0	1	-	RW	<a href="#">ICC_BPR0</a>	<a href="#">ICC_BPR0</a>

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

ICV\_BPR0 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\_BPR0 results in an access to [ICC\\_BPR0](#).

The minimum binary point value is derived from the number of implemented priority bits. The number of priority bits is IMPLEMENTATION DEFINED, and reported by [ICV\\_CTLR](#).PRIbits.

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value. On a reset, the binary point field is set to the minimum supported value.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# ICV\_BPR1, Interrupt Controller Virtual Binary Point Register 1

The ICV\_BPR1 characteristics are:

## Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines virtual Group 1 interrupt preemption.

This register is part of:

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

## Configuration

AArch32 System register ICV\_BPR1 is architecturally mapped to AArch64 System register [ICV\\_BPR1\\_EL1](#).

## Attributes

ICV\_BPR1 is a 32-bit register.

## Field descriptions

The ICV\_BPR1 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	0

BinaryPoint

### Bits [31:3]

Reserved, RES0.

### BinaryPoint, bits [2:0]

If the GIC is configured to use separate binary point fields for virtual Group 0 and virtual Group 1 interrupts, the value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. This is done as follows:

Binary point value	Group priority field	Subpriority field	Field with binary point
0	-	-	-
1	[7:1]	[0]	ggggggg.s
2	[7:2]	[1:0]	ggggggg.ss
3	[7:3]	[2:0]	ggggggg.sss
4	[7:4]	[3:0]	ggggg.ssss
5	[7:5]	[4:0]	ggg.sssss
6	[7:6]	[5:0]	gg.ssssss
7	[7]	[6:0]	g.sssssss

Writing 0 to this field will set this field to its reset value, which is IMPLEMENTATION DEFINED and non-zero.

If [ICV\\_CTLR](#).CBPR is set to 1, Non-secure EL1 reads return [ICV\\_BPR0](#) + 1 saturated to 0b111. Non-secure EL1 writes are ignored.

## Accessing the ICV\_BPR1

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, 3	000	011	1100	1111	1100

When HCR.IMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_BPR1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_BPR1</a>	n/a	<a href="#">ICC_BPR1</a>
x	x	1	1	-	n/a	<a href="#">ICC_BPR1</a>	<a href="#">ICC_BPR1</a>
x	0	0	1	-	<a href="#">ICC_BPR1</a>	<a href="#">ICC_BPR1</a>	<a href="#">ICC_BPR1</a>
x	1	0	1	-	RW	<a href="#">ICC_BPR1</a>	<a href="#">ICC_BPR1</a>

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

ICV\_BPR1 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\_BPR1 results in an access to [ICC\\_BPR1](#).

The reset value is IMPLEMENTATION DEFINED, but is equal to the minimum value of [ICV\\_BPR0](#) plus one.

An attempt to program the binary point field to a value less than the reset value sets the field to the reset value.

## 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 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

02/05/2017 15:43

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

# ICV\_CTLR, Interrupt Controller Virtual Control Register

The ICV\_CTLR characteristics are:

## Purpose

Controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

This register is part of:

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

## Configuration

AArch32 System register ICV\_CTLR is architecturally mapped to AArch64 System register [ICV\\_CTLR\\_EL1](#).

## Attributes

ICV\_CTLR is a 32-bit register.

## Field descriptions

The ICV\_CTLR 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	<a href="#">A3V</a>	<a href="#">SEIS</a>	<a href="#">IDbits</a>	<a href="#">PRIbits</a>	0	0	0	0	0	0	0	0	<a href="#">EOImode</a>	<a href="#">CBPR</a>		

### Bits [31:16]

Reserved, RES0.

### A3V, bit [15]

Affinity 3 Valid. Read-only and writes are ignored. Possible values are:

A3V	Meaning
0	The virtual CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers.
1	The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

### SEIS, bit [14]

SEI Support. Read-only and writes are ignored. Indicates whether the virtual CPU interface supports local generation of SEIs:

SEIS	Meaning
0	The virtual CPU interface logic does not support local generation of SEIs.
1	The virtual CPU interface logic supports local generation of SEIs.

### IDbits, bits [13:11]

Identifier bits. Read-only and writes are ignored. The number of virtual interrupt identifier bits supported:

IDbits	Meaning
000	16 bits.
001	24 bits.

All other values are reserved.

### PRibits, bits [10:8]

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation must implement at least 32 levels of physical priority (5 priority bits).

#### Note

This field always returns the number of priority bits implemented.

The division between group priority and subpriority is defined in the binary point registers [ICV\\_BPR0](#) and [ICV\\_BPR1](#).

### Bits [7:2]

Reserved, RES0.

### EOImode, bit [1]

Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt:

EOImode	Meaning
0	<a href="#">ICV_EOIR0</a> and <a href="#">ICV_EOIR1</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">ICV_DIR</a> are UNPREDICTABLE.
1	<a href="#">ICV_EOIR0</a> and <a href="#">ICV_EOIR1</a> provide priority drop functionality only. <a href="#">ICV_DIR</a> provides interrupt deactivation functionality.

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

### CBPR, bit [0]

Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts:

CBPR	Meaning
0	<a href="#">ICV_BPR0</a> determines the preemption group for virtual Group 0 interrupts only. <a href="#">ICV_BPR1</a> determines the preemption group for virtual Group 1 interrupts.
1	<a href="#">ICV_BPR0</a> determines the preemption group for both virtual Group 0 and virtual Group 1 interrupts. Reads of <a href="#">ICV_BPR1</a> return <a href="#">ICV_BPR0</a> plus one, saturated to 0b111. Writes to <a href="#">ICV_BPR1</a> are ignored.

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

## Accessing the ICV\_CTLR

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, 4	000	100	1100	1111	1100

When HCR.{FMO, IMO} == {0, 0}, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_CTLR](#).



## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_CTLR</a>	n/a	<a href="#">ICC_CTLR</a>
x	x	1	1	-	n/a	<a href="#">ICC_CTLR</a>	<a href="#">ICC_CTLR</a>
x	1	0	1	-	RW	<a href="#">ICC_CTLR</a>	<a href="#">ICC_CTLR</a>
1	x	0	1	-	RW	<a href="#">ICC_CTLR</a>	<a href="#">ICC_CTLR</a>
0	0	0	1	-	<a href="#">ICC_CTLR</a>	<a href="#">ICC_CTLR</a>	<a href="#">ICC_CTLR</a>

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

ICV\_CTLR is only accessible at Non-secure EL1 when  $HCR.\{FMO, IMO\} \neq \{0, 0\}$ .

### Note

When  $HCR.\{FMO, IMO\} = \{0, 0\}$ , at Non-secure EL1, the instruction encoding used to access ICV\_CTLR results in an access to [ICC\\_CTLR](#).

## 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  $SCR\_EL3.NS==1 \ \&\& \ .E2H==0$  :

- If [HSTR\\_EL2](#).T12==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and  $SCR\_EL3.NS==1 \ \&\& \ .E2H==1 \ \&\& \ HCR\_EL2.TGE==0$  :

- 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](#).TC==1, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure accesses to this register from EL1 are trapped to EL2.

# ICV\_DIR, Interrupt Controller Deactivate Virtual Interrupt Register

The ICV\_DIR characteristics are:

## Purpose

When interrupt priority drop is separated from interrupt deactivation, a write to this register deactivates the specified virtual interrupt.

This register is part of:

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

## Configuration

AArch32 System register ICV\_DIR performs the same function as AArch64 System register [ICV\\_DIR\\_EL1](#).

## Attributes

ICV\_DIR is a 32-bit register.

## Field descriptions

The ICV\_DIR 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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the virtual interrupt to be deactivated.

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

## Accessing the ICV\_DIR

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, c11, 1	000	001	1100	1111	1011

This encoding results in an access to ICV\_DIR at Non-secure EL1 in the following cases:

- When HCR.FMO is set to 1.
- When HCR.IMO is set to 1.

This encoding results in an access to [ICC\\_DIR](#) at Non-secure EL1 in the following cases:

- When [HCR2](#).{FMO, IMO} == {0, 0}.

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_DIR</a>	n/a	<a href="#">ICC_DIR</a>
x	x	1	1	-	n/a	<a href="#">ICC_DIR</a>	<a href="#">ICC_DIR</a>
x	1	0	1	-	WO	<a href="#">ICC_DIR</a>	<a href="#">ICC_DIR</a>
1	x	0	1	-	WO	<a href="#">ICC_DIR</a>	<a href="#">ICC_DIR</a>
0	0	0	1	-	<a href="#">ICC_DIR</a>	<a href="#">ICC_DIR</a>	<a href="#">ICC_DIR</a>

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

The ICV\_DIR register is only accessible at Non-secure EL1 in the following cases:

- When HCR.FMO is set to 1.
- When HCR.IMO is set to 1.

---

### Note

At Non-secure EL1, the instruction encoding used to access ICV\_DIR results in an access to [ICC\\_DIR](#) when HCR.{FMO, IMO} == {0, 0}.

---

When EOImode == 0, writes are ignored In systems supporting system error generation, an implementation might generate an SEI.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 write accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure write accesses to this register from EL1 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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 [HSTR](#).T12==1, Non-secure write accesses to this register from EL1 are trapped to Hyp mode.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR](#).TC==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

# ICV\_EOIR0, Interrupt Controller Virtual End Of Interrupt Register 0

The ICV\_EOIR0 characteristics are:

## Purpose

A PE writes to this register to inform the CPU interface that it has completed the processing of the specified virtual Group 0 interrupt.

This register is part of:

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

## Configuration

AArch32 System register ICV\_EOIR0 performs the same function as AArch64 System register [ICV\\_EOIR0\\_EL1](#).

## Attributes

ICV\_EOIR0 is a 32-bit register.

## Field descriptions

The ICV\_EOIR0 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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID from the corresponding [ICV\\_IAR0](#) access.

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

If the [ICV\\_CTLR](#).EOImode bit is 0, a write to this register drops the priority for the virtual interrupt, and also deactivates the virtual interrupt.

If the [ICV\\_CTLR](#).EOImode bit is 1, a write to this register only drops the priority for the virtual interrupt. Software must write to [ICV\\_DIR](#) to deactivate the virtual interrupt.

## Accessing the ICV\_EOIR0

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, c8, 1	000	001	1100	1111	1000

When HCR.FMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_EOIR0](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_EOIR0</a>	n/a	<a href="#">ICC_EOIR0</a>
x	x	1	1	-	n/a	<a href="#">ICC_EOIR0</a>	<a href="#">ICC_EOIR0</a>
0	x	0	1	-	<a href="#">ICC_EOIR0</a>	<a href="#">ICC_EOIR0</a>	<a href="#">ICC_EOIR0</a>
1	x	0	1	-	WO	<a href="#">ICC_EOIR0</a>	<a href="#">ICC_EOIR0</a>

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

ICV\_EOIR0 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\_EOIR0 results in an access to [ICC\\_EOIR0](#).

A write to this register must correspond to the most recent valid read by this vPE from a Virtual Interrupt Acknowledge Register, and must correspond to the INTID that was read from [ICV\\_IAR0](#), otherwise the system behavior is UNPREDICTABLE. A valid read is a read that returns a valid INTID that is not a special INTID.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 write accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure write accesses to this register from EL1 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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 [HSTR](#).T12==1, Non-secure write accesses to this register from EL1 are trapped to Hyp mode.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR](#).TALL0==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

# ICV\_EOIR1, Interrupt Controller Virtual End Of Interrupt Register 1

The ICV\_EOIR1 characteristics are:

## Purpose

A PE writes to this register to inform the CPU interface that it has completed the processing of the specified virtual Group 1 interrupt.

This register is part of:

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

## Configuration

AArch32 System register ICV\_EOIR1 performs the same function as AArch64 System register [ICV\\_EOIR1\\_EL1](#).

## Attributes

ICV\_EOIR1 is a 32-bit register.

## Field descriptions

The ICV\_EOIR1 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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID from the corresponding [ICV\\_IAR1](#) access.

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

If the [ICV\\_CTLR.EOImode](#) bit is 0, a write to this register drops the priority for the virtual interrupt, and also deactivates the virtual interrupt.

If the [ICV\\_CTLR.EOImode](#) bit is 1, a write to this register only drops the priority for the virtual interrupt. Software must write to [ICV\\_DIR](#) to deactivate the virtual interrupt.

## Accessing the ICV\_EOIR1

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, 1	000	001	1100	1111	1100

When HCR.IMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_EOIR1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_EOIR1</a>	n/a	<a href="#">ICC_EOIR1</a>
x	x	1	1	-	n/a	<a href="#">ICC_EOIR1</a>	<a href="#">ICC_EOIR1</a>
x	0	0	1	-	<a href="#">ICC_EOIR1</a>	<a href="#">ICC_EOIR1</a>	<a href="#">ICC_EOIR1</a>
x	1	0	1	-	WO	<a href="#">ICC_EOIR1</a>	<a href="#">ICC_EOIR1</a>

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

ICV\_EOIR1 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\_EOIR1 results in an access to [ICC\\_EOIR1](#).

A write to this register must correspond to the most recent valid read by this vPE from a Virtual Interrupt Acknowledge Register, and must correspond to the INTID that was read from [ICV\\_IAR1](#), otherwise the system behavior is UNPREDICTABLE. A valid read is a read that returns a valid INTID that is not a special INTID.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 write accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure write accesses to this register from EL1 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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 [HSTR](#).T12==1, Non-secure write accesses to this register from EL1 are trapped to Hyp mode.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR](#).TALL1==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

# ICV\_HPPIR0, Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0

The ICV\_HPPIR0 characteristics are:

## Purpose

Indicates the highest priority pending virtual Group 0 interrupt on the virtual CPU interface.

This register is part of:

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

## Configuration

AArch32 System register ICV\_HPPIR0 performs the same function as AArch64 System register [ICV\\_HPPIR0\\_EL1](#).

## Attributes

ICV\_HPPIR0 is a 32-bit register.

## Field descriptions

The ICV\_HPPIR0 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	INTID																							

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the highest priority pending virtual interrupt.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. This special INTID can take the value 1023 only. See Special INTIDs, for more information.

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

## Accessing the ICV\_HPPIR0

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>, c12, c8, 2	000	010	1100	1111	1000

When HCR.FMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_HPPIR0](#).



## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_HPPIR0</a>	n/a	<a href="#">ICC_HPPIR0</a>
x	x	1	1	-	n/a	<a href="#">ICC_HPPIR0</a>	<a href="#">ICC_HPPIR0</a>
0	x	0	1	-	<a href="#">ICC_HPPIR0</a>	<a href="#">ICC_HPPIR0</a>	<a href="#">ICC_HPPIR0</a>
1	x	0	1	-	RO	<a href="#">ICC_HPPIR0</a>	<a href="#">ICC_HPPIR0</a>

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

ICV\_HPPIR0 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\_HPPIR0 results in an access to [ICC\\_HPPIR0](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 read accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure read accesses to this register from EL1 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==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 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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](#).T12==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR](#).TALL0==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

# ICV\_HPPIR1, Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1

The ICV\_HPPIR1 characteristics are:

## Purpose

Indicates the highest priority pending virtual Group 1 interrupt on the virtual CPU interface.

This register is part of:

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

## Configuration

AArch32 System register ICV\_HPPIR1 performs the same function as AArch64 System register [ICV\\_HPPIR1\\_EL1](#).

## Attributes

ICV\_HPPIR1 is a 32-bit register.

## Field descriptions

The ICV\_HPPIR1 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	INTID																							

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the highest priority pending virtual interrupt.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. This special INTID can take the value 1023 only. See Special INTIDs, for more information.

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

## Accessing the ICV\_HPPIR1

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>, c12, c12, 2	000	010	1100	1111	1100

When HCR.IMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_HPPIR1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_HPPIR1</a>	n/a	<a href="#">ICC_HPPIR1</a>
x	x	1	1	-	n/a	<a href="#">ICC_HPPIR1</a>	<a href="#">ICC_HPPIR1</a>
x	0	0	1	-	<a href="#">ICC_HPPIR1</a>	<a href="#">ICC_HPPIR1</a>	<a href="#">ICC_HPPIR1</a>
x	1	0	1	-	RO	<a href="#">ICC_HPPIR1</a>	<a href="#">ICC_HPPIR1</a>

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

ICV\_HPPIR1 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\_HPPIR1 results in an access to [ICC\\_HPPIR1](#).

## 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 read accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure read accesses to this register from EL1 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==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 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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](#).T12==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR](#).TALL1==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

# ICV\_IAR0, Interrupt Controller Virtual Interrupt Acknowledge Register 0

The ICV\_IAR0 characteristics are:

## Purpose

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

This register is part of:

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

## Configuration

AArch32 System register ICV\_IAR0 performs the same function as AArch64 System register [ICV\\_IAR0\\_EL1](#).

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

## Attributes

ICV\_IAR0 is a 32-bit register.

## Field descriptions

The ICV\_IAR0 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													INTID											

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the signaled virtual interrupt.

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

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. This special INTID can take the value 1023 only. See Special INTIDs, for more information.

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

## Accessing the ICV\_IAR0

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>, c12, c8, 0	000	000	1100	1111	1000

When HCR.FMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_IAR0](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	<a href="#">ICC_IAR0</a>	<a href="#">ICC_IAR0</a>	n/a	<a href="#">ICC_IAR0</a>
x	x	1	1	-	n/a	<a href="#">ICC_IAR0</a>	<a href="#">ICC_IAR0</a>
0	x	0	1	-	<a href="#">ICC_IAR0</a>	<a href="#">ICC_IAR0</a>	<a href="#">ICC_IAR0</a>
1	x	0	1	-	RO	<a href="#">ICC_IAR0</a>	<a href="#">ICC_IAR0</a>

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

ICV\_IAR0 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\_IAR0 results in an access to [ICC\\_IAR0](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 read accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure read accesses to this register from EL1 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==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 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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](#).T12==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR](#).TALL0==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure read accesses to this register from EL1 are trapped to EL2.



# ICV\_IAR1, Interrupt Controller Virtual Interrupt Acknowledge Register 1

The ICV\_IAR1 characteristics are:

## Purpose

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

This register is part of:

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

## Configuration

AArch32 System register ICV\_IAR1 performs the same function as AArch64 System register [ICV\\_IAR1\\_EL1](#).

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

## Attributes

ICV\_IAR1 is a 32-bit register.

## Field descriptions

The ICV\_IAR1 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													INTID											

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the signaled virtual interrupt.

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

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. This special INTID can take the value 1023 only. See Special INTIDs, for more information.

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

## Accessing the ICV\_IAR1

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>, c12, c12, 0	000	000	1100	1111	1100

When HCR.IMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_IAR1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_IAR1</a>	n/a	<a href="#">ICC_IAR1</a>
x	x	1	1	-	n/a	<a href="#">ICC_IAR1</a>	<a href="#">ICC_IAR1</a>
x	0	0	1	-	<a href="#">ICC_IAR1</a>	<a href="#">ICC_IAR1</a>	<a href="#">ICC_IAR1</a>
x	1	0	1	-	RO	<a href="#">ICC_IAR1</a>	<a href="#">ICC_IAR1</a>

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

ICV\_IAR1 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\_IAR1 results in an access to [ICC\\_IAR1](#).

## 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 read accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure read accesses to this register from EL1 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HSTR\\_EL2](#).T12==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 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HSTR\\_EL2](#).T12==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](#).T12==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR](#).TALL1==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure read accesses to this register from EL1 are trapped to EL2.





# 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:

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

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<sup>®</sup> 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 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

# 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:

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

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<sup>®</sup> 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 && .E2H==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 AArch64 and SCR\_EL3.NS==1 && .E2H==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.

# ICV\_PMR, Interrupt Controller Virtual Interrupt Priority Mask Register

The ICV\_PMR characteristics are:

## Purpose

Provides a virtual interrupt priority filter. Only virtual interrupts with a higher priority than the value in this register are signaled to the PE.

This register is part of:

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

## Configuration

AArch32 System register ICV\_PMR is architecturally mapped to AArch64 System register [ICV\\_PMR\\_ELI](#).

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that writes to this register are self-synchronising. This ensures that no interrupts below the written PMR value will be taken after a write to this register is architecturally executed. See Observability of the effects of accesses to the GIC registers, for more information.

## Attributes

ICV\_PMR is a 32-bit register.

## Field descriptions

The ICV\_PMR 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	Priority							

### Bits [31:8]

Reserved, RES0.

### Priority, bits [7:0]

The priority mask level for the virtual CPU interface. If the priority of a virtual interrupt is higher than the value indicated by this field, the interface signals the virtual interrupt to the PE.

The possible priority field values are as follows:

Implemented priority bits	Possible priority field values	Number of priority levels
[7:0]	0x00-0xFF (0-255), all values	256
[7:1]	0x00-0xFE (0-254), even values only	128
[7:2]	0x00-0xFC (0-252), in steps of 4	64
[7:3]	0x00-0xF8 (0-248), in steps of 8	32
[7:4]	0x00-0xF0 (0-240), in steps of 16	16

Unimplemented priority bits are RAZ/WI.

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

## Accessing the ICV\_PMR

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>, c4, c6, 0	000	000	0100	1111	0110

When  $HCR.\{FMO, IMO\} == \{0, 0\}$ , execution of this encoding at Non-secure EL1 results in an access to [ICC\\_PMR](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_PMR</a>	n/a	<a href="#">ICC_PMR</a>
x	x	1	1	-	n/a	<a href="#">ICC_PMR</a>	<a href="#">ICC_PMR</a>
x	1	0	1	-	RW	<a href="#">ICC_PMR</a>	<a href="#">ICC_PMR</a>
1	x	0	1	-	RW	<a href="#">ICC_PMR</a>	<a href="#">ICC_PMR</a>
0	0	0	1	-	<a href="#">ICC_PMR</a>	<a href="#">ICC_PMR</a>	<a href="#">ICC_PMR</a>

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

ICV\_PMR is only accessible at Non-secure EL1 when  $HCR.\{FMO, IMO\} \neq \{0, 0\}$ .

### Note

When  $HCR.\{FMO, IMO\} == \{0, 0\}$ , at Non-secure EL1, the instruction encoding used to access ICV\_PMR results in an access to [ICC\\_PMR](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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  $SCR\_EL3.NS==1$  :

- If [ICH\\_HCR](#).TC==1, Non-secure accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure accesses to this register from EL1 are trapped to EL2.

# ICV\_RPR, Interrupt Controller Virtual Running Priority Register

The ICV\_RPR characteristics are:

## Purpose

Indicates the Running priority of the virtual CPU interface.

This register is part of:

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

## Configuration

AArch32 System register ICV\_RPR performs the same function as AArch64 System register [ICV\\_RPR\\_EL1](#).

## Attributes

ICV\_RPR is a 32-bit register.

## Field descriptions

The ICV\_RPR 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	Priority							

### Bits [31:8]

Reserved, RES0.

### Priority, bits [7:0]

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

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

---

#### Note

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

---

## Accessing the ICV\_RPR

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>, c12, c11, 3	000	011	1100	1111	1011

When HCR.{FMO, IMO} == {0, 0}, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_RPR](#).



## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_RPR</a>	n/a	<a href="#">ICC_RPR</a>
x	x	1	1	-	n/a	<a href="#">ICC_RPR</a>	<a href="#">ICC_RPR</a>
x	1	0	1	-	RO	<a href="#">ICC_RPR</a>	<a href="#">ICC_RPR</a>
1	x	0	1	-	RO	<a href="#">ICC_RPR</a>	<a href="#">ICC_RPR</a>
0	0	0	1	-	<a href="#">ICC_RPR</a>	<a href="#">ICC_RPR</a>	<a href="#">ICC_RPR</a>

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

ICV\_RPR is only accessible at Non-secure EL1 when  $HCR.\{FMO, IMO\} \neq \{0, 0\}$ .

### Note

When  $HCR.\{FMO, IMO\} = \{0, 0\}$ , at Non-secure EL1, the instruction encoding used to access ICV\_RPR results in an access to [ICC\\_RPR](#).

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

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

## 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 read accesses to this register from EL1 are UNDEFINED.
- If [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure read accesses to this register from EL1 are trapped to EL1.

When EL2 is implemented and is using AArch64 and  $SCR\_EL3.NS=1 \ \&\& \ .E2H=0$  :

- If [HSTR\\_EL2](#).T12==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and  $SCR\_EL3.NS=1 \ \&\& \ .E2H=1 \ \&\& \ HCR\_EL2.TGE=0$  :

- If [HSTR\\_EL2](#).T12==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](#).T12==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When  $SCR\_EL3.NS=1$  :

- If [ICH\\_HCR](#).TC==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

# ID\_AFR0, Auxiliary Feature Register 0

The ID\_AFR0 characteristics are:

## Purpose

Provides information about the IMPLEMENTATION DEFINED features of the PE in AArch32 state.

Must be interpreted with the Main ID Register, [MIDR](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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\_AFR0 is architecturally mapped to AArch64 System register [ID\\_AFR0\\_EL1](#).

## Attributes

ID\_AFR0 is a 32-bit register.

## Field descriptions

The ID\_AFR0 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	IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED			

### Bits [31:16]

Reserved, RES0.

### IMPLEMENTATION DEFINED, bits [15:12]

IMPLEMENTATION DEFINED.

### IMPLEMENTATION DEFINED, bits [11:8]

IMPLEMENTATION DEFINED.

### IMPLEMENTATION DEFINED, bits [7:4]

IMPLEMENTATION DEFINED.

### IMPLEMENTATION DEFINED, bits [3:0]

IMPLEMENTATION DEFINED.

## Accessing the ID\_AFR0

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, 3	000	011	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.

# ID\_DFR0, Debug Feature Register 0

The ID\_DFR0 characteristics are:

## Purpose

Provides top level information about the debug system in AArch32 state.

Must be interpreted with the Main ID Register, [MIDR](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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\_DFR0 is architecturally mapped to AArch64 System register [ID\\_DFR0\\_EL1](#).

## Attributes

ID\_DFR0 is a 32-bit register.

## Field descriptions

The ID\_DFR0 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		<a href="#">PerfMon</a>			<a href="#">MProfDbg</a>				<a href="#">MMapTrc</a>				<a href="#">CopTrc</a>				<a href="#">MMapDbg</a>			<a href="#">CopSDBG</a>				<a href="#">CopDbg</a>				

### Bits [31:28]

Reserved, RES0.

### PerfMon, bits [27:24]

Performance Monitors. Support for System registers-based ARM Performance Monitors Extension, using registers in the coproc == 1111 encoding space, for A and R profile processors. Defined values are:

PerfMon	Meaning
0000	Performance Monitors Extension System registers not implemented.
0001	Support for Performance Monitors Extension version 1 (PMUv1) System registers.
0010	Support for Performance Monitors Extension version 2 (PMUv2) System registers.
0011	Support for Performance Monitors Extension version 3 (PMUv3) System registers.
0100	Support for Performance Monitors Extension version 3 (PMUv3) System registers, with a 16-bit evtCount field.
1111	IMPLEMENTATION DEFINED form of Performance Monitors System registers supported. PMUv3 not supported.

All other values are reserved.

In ARMv8.0 the permitted values are 0000, 0011, and 1111.

From ARMv8.1 the permitted values are 0000, 0100, and 1111.

In ARMv7, the value 0000 can mean that PMUv1 is implemented. PMUv1 is not permitted in an ARMv8 implementation.

### MProfDbg, bits [23:20]

M Profile Debug. Support for memory-mapped debug model for M profile processors. Defined values are:

MProfDbg	Meaning
0000	Not supported.
0001	Support for M profile Debug architecture, with memory-mapped access.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### MMapTrc, bits [19:16]

Memory Mapped Trace. Support for memory-mapped trace model. Defined values are:

MMapTrc	Meaning
0000	Not supported.
0001	Support for ARM trace architecture, with memory-mapped access.

All other values are reserved.

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

In the Trace registers, the ETMIDR gives more information about the implementation.

### CopTrc, bits [15:12]

Support for System registers-based trace model, using registers in the coproc == 1110 encoding space. Defined values are:

CopTrc	Meaning
0000	Not supported.
0001	Support for ARM trace architecture, with System registers access.

All other values are reserved.

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

In the Trace registers, the ETMIDR gives more information about the implementation.

### MMapDbg, bits [11:8]

Memory Mapped Debug. Support for v7 memory-mapped debug model, for A and R profile processors.

In ARMv8-A this field is RES0.

The optional memory map defined by ARMv8 is not compatible with ARMv7.

### CopSDBG, bits [7:4]

Support for a System registers-based Secure debug model, using registers in the coproc = 1110 encoding space, for an A profile processor that includes EL3.

If EL3 is not implemented and the implemented Security state is Non-Secure state, this field is RES0. Otherwise, this field reads the same as bits [3:0].

### CopDbg, bits [3:0]

Support for System registers-based debug model, using registers in the coproc == 1110 encoding space, for A and R profile processors. Defined values are:

CopDbg	Meaning
0000	Not supported.
0010	Support for ARMv6, v6 Debug architecture, with System registers access.
0011	Support for ARMv6, v6.1 Debug architecture, with System registers access.
0100	Support for ARMv7, v7 Debug architecture, with System registers access.
0101	Support for ARMv7, v7.1 Debug architecture, with System registers access.
0110	Support for ARMv8 debug architecture, with System registers access.
0111	Support for ARMv8 debug architecture, with System registers access, and Virtualization Host extensions.
1000	Support for ARMv8.2 debug architecture.

All other values are reserved.

In an ARMv8.0 implementation, the only permitted value is 0b0110.

In an ARMv8.1 implementation that does not include ARMv8.1-VHE, the permitted values are 0b0110 and 0b0111.

In an ARMv8.1 implementation that includes ARMv8.1-VHE, the only permitted value is 0b0111.

In an ARMv8.2 implementation, the only permitted value is 0b1000.

## Accessing the ID\_DFR0

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, 2	000	010	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<sup>®</sup> 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.

02/05/2017 15:43

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

# ID\_ISAR0, Instruction Set Attribute Register 0

The ID\_ISAR0 characteristics are:

## Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID\\_ISAR1](#), [ID\\_ISAR2](#), [ID\\_ISAR3](#), [ID\\_ISAR4](#), and [ID\\_ISAR5](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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\_ISAR0 is architecturally mapped to AArch64 System register [ID\\_ISAR0\\_EL1](#).

## Attributes

ID\_ISAR0 is a 32-bit register.

## Field descriptions

The ID\_ISAR0 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		Divide			Debug				Coproc				CmpBranch				BitField				BitCount				Swap			

### Bits [31:28]

Reserved, RES0.

### Divide, bits [27:24]

Indicates the implemented Divide instructions. Defined values are:

Divide	Meaning
0000	None implemented.
0001	Adds SDIV and UDIV in the T32 instruction set.
0010	As for 0001, and adds SDIV and UDIV in the A32 instruction set.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

### Debug, bits [23:20]

Indicates the implemented Debug instructions. Defined values are:

Debug	Meaning
0000	None implemented.
0001	Adds BKPT.

All other values are reserved.



In ARMv8-A the only permitted value is 0001.

### Coproc, bits [19:16]

Indicates the implemented System register access instructions. Defined values are:

Coproc	Meaning
0000	None implemented, except for instructions separately attributed by the architecture to provide access to AArch32 System registers and System instructions.
0001	Adds generic CDP, LDC, MCR, MRC, and STC.
0010	As for 0001, and adds generic CDP2, LDC2, MCR2, MRC2, and STC2.
0011	As for 0010, and adds generic MCRR and MRRC.
0100	As for 0011, and adds generic MCRR2 and MRRC2.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### CmpBranch, bits [15:12]

Indicates the implemented combined Compare and Branch instructions in the T32 instruction set. Defined values are:

CmpBranch	Meaning
0000	None implemented.
0001	Adds CBNZ and CBZ.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### BitField, bits [11:8]

Indicates the implemented BitField instructions. Defined values are:

BitField	Meaning
0000	None implemented.
0001	Adds BFC, BFI, SBFX, and UBFX.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### BitCount, bits [7:4]

Indicates the implemented Bit Counting instructions. Defined values are:

BitCount	Meaning
0000	None implemented.
0001	Adds CLZ.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Swap, bits [3:0]

Indicates the implemented Swap instructions in the A32 instruction set. Defined values are:

Swap	Meaning
0000	None implemented.
0001	Adds SWP and SWPB.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

## Accessing the ID\_ISAR0

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, c2, 0	000	000	0000	1111	0010

## 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<sup>®</sup> 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.

# ID\_ISAR1, Instruction Set Attribute Register 1

The ID\_ISAR1 characteristics are:

## Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID\\_ISAR0](#), [ID\\_ISAR2](#), [ID\\_ISAR3](#), [ID\\_ISAR4](#), and [ID\\_ISAR5](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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\_ISAR1 is architecturally mapped to AArch64 System register [ID\\_ISAR1\\_EL1](#).

## Attributes

ID\_ISAR1 is a 32-bit register.

## Field descriptions

The ID\_ISAR1 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
<a href="#">Jazelle</a>	<a href="#">Interwork</a>	<a href="#">Immediate</a>	<a href="#">IfThen</a>	<a href="#">Extend</a>	<a href="#">Except_AR</a>	<a href="#">Except</a>	<a href="#">Endian</a>																								

### Jazelle, bits [31:28]

Indicates the implemented Jazelle extension instructions. Defined values are:

Jazelle	Meaning
0000	No support for Jazelle.
0001	Adds the BXJ instruction, and the J bit in the PSR. This setting might indicate a trivial implementation of the Jazelle extension.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Interwork, bits [27:24]

Indicates the implemented Interworking instructions. Defined values are:

Interwork	Meaning
0000	None implemented.
0001	Adds the BX instruction, and the T bit in the PSR.
0010	As for 0001, and adds the BLX instruction. PC loads have BX-like behavior.
0011	As for 0010, and guarantees that data-processing instructions in the A32 instruction set with the PC as the destination and the S bit clear have BX-like behavior.

All other values are reserved.

In ARMv8-A the only permitted value is 0011.

### Immediate, bits [23:20]

Indicates the implemented data-processing instructions with long immediates. Defined values are:

Immediate	Meaning
0000	None implemented.
0001	Adds: <ul style="list-style-type: none"> <li>The MOVT instruction.</li> <li>The MOV instruction encodings with zero-extended 16-bit immediates.</li> <li>The T32 ADD and SUB instruction encodings with zero-extended 12-bit immediates, and the other ADD, ADR, and SUB encodings cross-referenced by the pseudocode for those encodings.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### IfThen, bits [19:16]

Indicates the implemented If-Then instructions in the T32 instruction set. Defined values are:

IfThen	Meaning
0000	None implemented.
0001	Adds the IT instructions, and the IT bits in the PSRs.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Extend, bits [15:12]

Indicates the implemented Extend instructions. Defined values are:

Extend	Meaning
0000	No scalar sign-extend or zero-extend instructions are implemented, where scalar instructions means non-Advanced SIMD instructions.
0001	Adds the SXTB, SXTB, UXTB, and UXTB instructions.
0010	As for 0001, and adds the SXTB16, SXTAB, SXTAB16, SXTAH, UXTB16, UXTAB, UXTAB16, and UXTAH instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

### Except\_AR, bits [11:8]

Indicates the implemented A and R profile exception-handling instructions. Defined values are:

Except_AR	Meaning
0000	None implemented.
0001	Adds the SRS and RFE instructions, and the A and R profile forms of the CPS instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Except, bits [7:4]

Indicates the implemented exception-handling instructions in the ARM instruction set. Defined values are:

Except	Meaning
0000	Not implemented. This indicates that the User bank and Exception return forms of the LDM and STM instructions are not implemented.
0001	Adds the LDM (exception return), LDM (user registers), and STM (user registers) instruction versions.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Endian, bits [3:0]

Indicates the implemented Endian instructions. Defined values are:

Endian	Meaning
0000	None implemented.
0001	Adds the SETEND instruction, and the E bit in the PSRs.

All other values are reserved.

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

## Accessing the ID\_ISAR1

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, c2, 1	000	001	0000	1111	0010

## 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<sup>®</sup> 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.

02/05/2017 15:43

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

# ID\_ISAR2, Instruction Set Attribute Register 2

The ID\_ISAR2 characteristics are:

## Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID\\_ISAR0](#), [ID\\_ISAR1](#), [ID\\_ISAR3](#), [ID\\_ISAR4](#), and [ID\\_ISAR5](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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\_ISAR2 is architecturally mapped to AArch64 System register [ID\\_ISAR2\\_EL1](#).

## Attributes

ID\_ISAR2 is a 32-bit register.

## Field descriptions

The ID\_ISAR2 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
Reversal				PSR_AR				MultU				MultS				Mult				MultiAccessInt				MemHint				LoadStore			

### Reversal, bits [31:28]

Indicates the implemented Reversal instructions. Defined values are:

Reversal	Meaning
0000	None implemented.
0001	Adds the REV, REV16, and REVSH instructions.
0010	As for 0001, and adds the RBIT instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

### PSR\_AR, bits [27:24]

Indicates the implemented A and R profile instructions to manipulate the PSR. Defined values are:

PSR_AR	Meaning
0000	None implemented.
0001	Adds the MRS and MSR instructions, and the exception return forms of data-processing instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

The exception return forms of the data-processing instructions are:

- In the A32 instruction set, data-processing instructions with the PC as the destination and the S bit set. These instructions might be affected by the WithShifts attribute.
- In the T32 instruction set, the SUBS PC,LR,#N instruction.

**MultU, bits [23:20]**

Indicates the implemented advanced unsigned Multiply instructions. Defined values are:

<b>MultU</b>	<b>Meaning</b>
0000	None implemented.
0001	Adds the UMULL and UMLAL instructions.
0010	As for 0001, and adds the UMAAL instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

**MultS, bits [19:16]**

Indicates the implemented advanced signed Multiply instructions. Defined values are:

<b>MultS</b>	<b>Meaning</b>
0000	None implemented.
0001	Adds the SMULL and SMLAL instructions.
0010	As for 0001, and adds the SMLABB, SMLABT, SMLALBB, SMLALBT, SMLALTB, SMLALTT, SMLATB, SMLATT, SMLAWB, SMLAWT, SMULBB, SMULBT, SMULTB, SMULTT, SMULWB, and SMULWT instructions. Also adds the Q bit in the PSRs.
0011	As for 0010, and adds the SMLAD, SMLADX, SMLALD, SMLALDX, SMLSD, SMLSDX, SMLSLD, SMLSLDX, SMMLA, SMMLAR, SMMLS, SMMLSR, SMMUL, SMMULR, SMUAD, SMUADX, SMUSD, and SMUSDX instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0011.

**Mult, bits [15:12]**

Indicates the implemented additional Multiply instructions. Defined values are:

<b>Mult</b>	<b>Meaning</b>
0000	No additional instructions implemented. This means only MUL is implemented.
0001	Adds the MLA instruction.
0010	As for 0001, and adds the MLS instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

**MultiAccessInt, bits [11:8]**

Indicates the support for interruptible multi-access instructions. Defined values are:

<b>MultiAccessInt</b>	<b>Meaning</b>
0000	No support. This means the LDM and STM instructions are not interruptible.
0001	LDM and STM instructions are restartable.
0010	LDM and STM instructions are continuable.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.



**MemHint, bits [7:4]**

Indicates the implemented Memory Hint instructions. Defined values are:

MemHint	Meaning
0000	None implemented.
0001	Adds the PLD instruction.
0010	Adds the PLD instruction. (0001 and 0010 have identical effects.)
0011	As for 0001 (or 0010), and adds the PLI instruction.
0100	As for 0011, and adds the PLDW instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0100.

**LoadStore, bits [3:0]**

Indicates the implemented additional load/store instructions. Defined values are:

LoadStore	Meaning
0000	No additional load/store instructions implemented.
0001	Adds the LDRD and STRD instructions.
0010	As for 0001, and adds the Load Acquire (LDAB, LDAH, LDA, LDAEXB, LDAEXH, LDAEX, LDAEXD) and Store Release (STLB, STLH, STL, STLEXB, STLEXH, STLEX, STLEXD) instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

**Accessing the ID\_ISAR2**

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, c2, 2	000	010	0000	1111	0010

**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<sup>®</sup> 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.

02/05/2017 15:43

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

# ID\_ISAR3, Instruction Set Attribute Register 3

The ID\_ISAR3 characteristics are:

## Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID\\_ISAR0](#), [ID\\_ISAR1](#), [ID\\_ISAR2](#), [ID\\_ISAR4](#), and [ID\\_ISAR5](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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\_ISAR3 is architecturally mapped to AArch64 System register [ID\\_ISAR3\\_ELI](#).

## Attributes

ID\_ISAR3 is a 32-bit register.

## Field descriptions

The ID\_ISAR3 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
<a href="#">T32EE</a>				<a href="#">TrueNOP</a>				<a href="#">T32Copy</a>				<a href="#">TabBranch</a>				<a href="#">SynchPrim</a>				<a href="#">SVC</a>				<a href="#">SIMD</a>				<a href="#">Saturate</a>			

### T32EE, bits [31:28]

Indicates the implemented T32EE instructions. Defined values are:

T32EE	Meaning
0000	None implemented.
0001	Adds the ENTERX and LEAVEX instructions, and modifies the load behavior to include null checking.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### TrueNOP, bits [27:24]

Indicates the implemented true NOP instructions. Defined values are:

TrueNOP	Meaning
0000	None implemented. This means there are no NOP instructions that do not have any register dependencies.
0001	Adds true NOP instructions in both the T32 and A32 instruction sets. This also permits additional NOP-compatible hints.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**T32Copy, bits [23:20]**

Indicates the support for T32 non flag-setting MOV instructions. Defined values are:

<b>T32Copy</b>	<b>Meaning</b>
0000	Not supported. This means that in the T32 instruction set, encoding T1 of the MOV (register) instruction does not support a copy from a low register to a low register.
0001	Adds support for T32 instruction set encoding T1 of the MOV (register) instruction, copying from a low register to a low register.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**TabBranch, bits [19:16]**

Indicates the implemented Table Branch instructions in the T32 instruction set. Defined values are:

<b>TabBranch</b>	<b>Meaning</b>
0000	None implemented.
0001	Adds the TBB and TBH instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**SynchPrim, bits [15:12]**

Used in conjunction with ID\_ISAR4.SynchPrim\_frac to indicate the implemented Synchronization Primitive instructions. Defined values are:

<b>SynchPrim</b>	<b>Meaning</b>
0000	If SynchPrim_frac == 0000, no Synchronization Primitives implemented.
0001	If SynchPrim_frac == 0000, adds the LDREX and STREX instructions.
	If SynchPrim_frac == 0011, also adds the CLREX, LDREXB, STREXB, and STREXH instructions.
0010	If SynchPrim_frac == 0000, as for [0001, 0011] and also adds the LDREXD and STREXD instructions.

All other combinations of SynchPrim and SynchPrim\_frac are reserved.

In ARMv8-A the only permitted value is 0010.

**SVC, bits [11:8]**

Indicates the implemented SVC instructions. Defined values are:

<b>SVC</b>	<b>Meaning</b>
0000	Not implemented.
0001	Adds the SVC instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**SIMD, bits [7:4]**

Indicates the implemented SIMD instructions. Defined values are:

SIMD	Meaning
0000	None implemented.
0001	Adds the SSAT and USAT instructions, and the Q bit in the PSRs.
0011	As for 0001, and adds the PKHBT, PKHTB, QADD16, QADD8, QASX, QSUB16, QSUB8, QSAX, SADD16, SADD8, SASX, SEL, SHADD16, SHADD8, SHASX, SHSUB16, SHSUB8, SHSAX, SSAT16, SSUB16, SSUB8, SSAX, SXTAB16, SXTB16, UADD16, UADD8, UASX, UHADD16, UHADD8, UHASX, UHSUB16, UHSUB8, UHSAX, UQADD16, UQADD8, UQASX, UQSUB16, UQSUB8, UQSAX, USAD8, USADA8, USAT16, USUB16, USUB8, USAX, UXTAB16, and UXTB16 instructions. Also adds support for the GE[3:0] bits in the PSRs.

All other values are reserved.

In ARMv8-A the only permitted value is 0011.

The SIMD field relates only to implemented instructions that perform SIMD operations on the general-purpose registers. In an implementation that supports floating-point and Advanced SIMD instructions, [MVFR0](#), [MVFR1](#), and [MVFR2](#) give information about the implemented Advanced SIMD instructions.

### Saturate, bits [3:0]

Indicates the implemented Saturate instructions. Defined values are:

Saturate	Meaning
0000	None implemented. This means no non-Advanced SIMD saturate instructions are implemented.
0001	Adds the QADD, QDADD, QDSUB, and QSUB instructions, and the Q bit in the PSRs.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

## Accessing the ID\_ISAR3

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, c2, 3	000	011	0000	1111	0010

## 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.

02/05/2017 15:43

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

# ID\_ISAR4, Instruction Set Attribute Register 4

The ID\_ISAR4 characteristics are:

## Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID\\_ISAR0](#), [ID\\_ISAR1](#), [ID\\_ISAR2](#), [ID\\_ISAR3](#), and [ID\\_ISAR5](#).

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\_ISAR4 is architecturally mapped to AArch64 System register [ID\\_ISAR4\\_EL1](#).

## Attributes

ID\_ISAR4 is a 32-bit register.

## Field descriptions

The ID\_ISAR4 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
SWP_frac				PSR_M				SynchPrim_frac				Barrier				SMC				Writeback				WithShifts				Unpriv			

### SWP\_frac, bits [31:28]

Indicates support for the memory system locking the bus for SWP or SWPB instructions. Defined values are:

SWP_frac	Meaning
0000	SWP or SWPB instructions not implemented.
0001	SWP or SWPB implemented but only in a uniprocessor context. SWP and SWPB do not guarantee whether memory accesses from other masters can come between the load memory access and the store memory access of the SWP or SWPB.

All other values are reserved. This field is valid only if the [ID\\_ISAR0](#).Swap\_instrs field is 0000.

In ARMv8-A the only permitted value is 0000.

### PSR\_M, bits [27:24]

Indicates the implemented M profile instructions to modify the PSRs. Defined values are:

PSR_M	Meaning
0000	None implemented.
0001	Adds the M profile forms of the CPS, MRS, and MSR instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

**SynchPrim\_frac, bits [23:20]**

Used in conjunction with [ID\\_ISAR3.SynchPrim](#) to indicate the implemented Synchronization Primitive instructions. Possible values are:

SynchPrim_frac	Meaning
0000	If SynchPrim == 0000, no Synchronization Primitives implemented. If SynchPrim == 0001, adds the LDREX and STREX instructions. If SynchPrim == 0010, also adds the CLREX, LDREXB, LDREXH, STREXB, STREXH, LDREXD, and STREXD instructions.
0011	If SynchPrim == 0001, adds the LDREX, STREX, CLREX, LDREXB, LDREXH, STREXB, and STREXH instructions.

All other combinations of SynchPrim and SynchPrim\_frac are reserved.

In ARMv8-A the only permitted value is 0000.

**Barrier, bits [19:16]**

Indicates the implemented Barrier instructions in the A32 and T32 instruction sets. Defined values are:

Barrier	Meaning
0000	None implemented. Barrier operations are provided only as System instructions in the (coproc==1111) encoding space.
0001	Adds the DMB, DSB, and ISB barrier instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**SMC, bits [15:12]**

Indicates the implemented SMC instructions. Defined values are:

SMC	Meaning
0000	None implemented.
0001	Adds the SMC instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**Writeback, bits [11:8]**

Indicates the support for Writeback addressing modes. Defined values are:

Writeback	Meaning
0000	Basic support. Only the LDM, STM, PUSH, POP, SRS, and RFE instructions support writeback addressing modes. These instructions support all of their writeback addressing modes.
0001	Adds support for all of the writeback addressing modes.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**WithShifts, bits [7:4]**

Indicates the support for instructions with shifts. Defined values are:

WithShifts	Meaning
0000	Nonzero shifts supported only in MOV and shift instructions.
0001	Adds support for shifts of loads and stores over the range LSL 0-3.
0011	As for 0001, and adds support for other constant shift options, both on load/store and other instructions.
0100	As for 0011, and adds support for register-controlled shift options.



All other values are reserved.

In ARMv8-A the only permitted value is 0100.

### Unpriv, bits [3:0]

Indicates the implemented unprivileged instructions. Defined values are:

Unpriv	Meaning
0000	None implemented. No T variant instructions are implemented.
0001	Adds the LDRBT, LDRT, STRBT, and STRT instructions.
0010	As for 0001, and adds the LDRHT, LDRSBT, LDRSHT, and STRHT instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

## Accessing the ID\_ISAR4

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, c2, 4	000	100	0000	1111	0010

## 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.

02/05/2017 15:43

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

# ID\_ISAR5, Instruction Set Attribute Register 5

The ID\_ISAR5 characteristics are:

## Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID\\_ISAR0](#), [ID\\_ISAR1](#), [ID\\_ISAR2](#), [ID\\_ISAR3](#), and [ID\\_ISAR4](#).

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\_ISAR5 is architecturally mapped to AArch64 System register [ID\\_ISAR5\\_ELI](#).

## Attributes

ID\_ISAR5 is a 32-bit register.

## Field descriptions

The ID\_ISAR5 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		RDM			0	0	0	0		CRC32				SHA2				SHA1				AES				SEVL		

### Bits [31:28]

Reserved, RES0.

### RDM, bits [27:24]

In ARMv8.2 and ARMv8.1:

Indicates whether the VQRDMLAH and VQRDMLSH instructions are implemented in AArch32 state. Defined values are:

RDM	Meaning
0000	No VQRDMLAH and VQRDMLSH instructions implemented.
0001	VQRDMLAH and VQRDMLSH instructions implemented.

All other values are reserved.

In ARMv8.0 the only permitted value is 0000.

From ARMv8.1 the only permitted value is 0001. This feature is identified by the name ARMv8.1-RDMA.

### In ARMv8.0:

Reserved, RES0.

**Bits [23:20]**

Reserved, RES0.

**CRC32, bits [19:16]**

Indicates whether the CRC32 instructions are implemented in AArch32 state.

CRC32	Meaning
0000	No CRC32 instructions implemented.
0001	CRC32B, CRC32H, CRC32W, CRC32CB, CRC32CH, and CRC32CW instructions implemented.

All other values are reserved.

In ARMv8.0 the permitted values are 0000 and 0001.

From ARMv8.1 the only permitted value is 0001.

**SHA2, bits [15:12]**

Indicates whether the SHA2 instructions are implemented in AArch32 state.

SHA2	Meaning
0000	No SHA2 instructions implemented.
0001	SHA256H, SHA256H2, SHA256SU0, and SHA256SU1 implemented.

All other values are reserved.

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

**SHA1, bits [11:8]**

Indicates whether the SHA1 instructions are implemented in AArch32 state.

SHA1	Meaning
0000	No SHA1 instructions implemented.
0001	SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 implemented.

All other values are reserved.

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

**AES, bits [7:4]**

Indicates whether the AES instructions are implemented in AArch32 state.

AES	Meaning
0000	No AES instructions implemented.
0001	AESE, AESD, AESMC, and AESIMC implemented.
0010	As for 0001, plus PMULL/PMULL2 instructions operating on 64-bit data quantities.

All other values are reserved.

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

**SEVL, bits [3:0]**

Indicates whether the SEVL instruction is implemented in AArch32 state.

SEVL	Meaning
0000	SEVL is implemented as a NOP.
0001	SEVL is implemented as Send Event Local.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

## Accessing the ID\_ISAR5

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, c2, 5	000	101	0000	1111	0010

## 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.

# ID\_MMFR0, Memory Model Feature Register 0

The ID\_MMFR0 characteristics are:

## Purpose

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

Must be interpreted with [ID\\_MMFR1](#), [ID\\_MMFR2](#), [ID\\_MMFR3](#), 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\_MMFR0 is architecturally mapped to AArch64 System register [ID\\_MMFR0\\_EL1](#).

## Attributes

ID\_MMFR0 is a 32-bit register.

## Field descriptions

The ID\_MMFR0 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
<a href="#">InnerShr</a>				<a href="#">FCSE</a>				<a href="#">AuxReg</a>				<a href="#">TCM</a>				<a href="#">ShareLvl</a>				<a href="#">OuterShr</a>				<a href="#">PMSA</a>				<a href="#">VMSA</a>			

### InnerShr, bits [31:28]

Innermost Shareability. Indicates the innermost shareability domain implemented. Defined values are:

InnerShr	Meaning
0000	Implemented as Non-cacheable.
0001	Implemented with hardware coherency support.
1111	Shareability ignored.

All other values are reserved.

In ARMv8 the permitted values are 0000, 0001, and 1111.

This field is valid only if the implementation supports two levels of shareability, as indicated by ID\_MMFR0.ShareLvl having the value 0001.

When ID\_MMFR0.ShareLvl is zero, this field is UNK.

### FCSE, bits [27:24]

Indicates whether the implementation includes the FCSE. Defined values are:

FCSE	Meaning
0000	Not supported.
0001	Support for FCSE.

All other values are reserved.

In ARMv8 the only permitted value is 0000.

### AuxReg, bits [23:20]

Auxiliary Registers. Indicates support for Auxiliary registers. Defined values are:

AuxReg	Meaning
0000	None supported.
0001	Support for Auxiliary Control Register only.
0010	Support for Auxiliary Fault Status Registers ( <a href="#">AIFSR</a> and <a href="#">ADEFSR</a> ) and Auxiliary Control Register.

All other values are reserved.

In ARMv8 the only permitted value is 0010.

#### Note

Accesses to unimplemented Auxiliary registers are UNDEFINED.

### TCM, bits [19:16]

Indicates support for TCMs and associated DMAs. Defined values are:

TCM	Meaning
0000	Not supported.
0001	Support is IMPLEMENTATION DEFINED. ARMv7 requires this setting.
0010	Support for TCM only, ARMv6 implementation.
0011	Support for TCM and DMA, ARMv6 implementation.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### ShareLvl, bits [15:12]

Shareability Levels. Indicates the number of shareability levels implemented. Defined values are:

ShareLvl	Meaning
0000	One level of shareability implemented.
0001	Two levels of shareability implemented.

All other values are reserved.

In ARMv8 the only permitted value is 0001.

### OuterShr, bits [11:8]

Outermost Shareability. Indicates the outermost shareability domain implemented. Defined values are:

OuterShr	Meaning
0000	Implemented as Non-cacheable.
0001	Implemented with hardware coherency support.
1111	Shareability ignored.

All other values are reserved.

In ARMv8 the permitted values are 0000, 0001, and 1111.

### PMSA, bits [7:4]

Indicates support for a PMSA. Defined values are:

PMSA	Meaning
0000	Not supported.
0001	Support for IMPLEMENTATION DEFINED PMSA.
0010	Support for PMSAv6, with a Cache Type Register implemented.
0011	Support for PMSAv7, with support for memory subsections. ARMv7-R profile.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### VMSA, bits [3:0]

Indicates support for a VMSA. Defined values are:

VMSA	Meaning
0000	Not supported.
0001	Support for IMPLEMENTATION DEFINED VMSA.
0010	Support for VMSAv6, with Cache and TLB Type Registers implemented.
0011	Support for VMSAv7, with support for remapping and the Access flag. ARMv7-A profile.
0100	As for 0011, and adds support for the PXN bit in the Short-descriptor translation table format descriptors.
0101	As for 0100, and adds support for the Long-descriptor translation table format.

All other values are reserved.

In ARMv8-A the only permitted value is 0101.

## Accessing the ID\_MMFR0

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, 4	000	100	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.

02/05/2017 15:43

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

# ID\_MMFR1, Memory Model Feature Register 1

The ID\_MMFR1 characteristics are:

## Purpose

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

Must be interpreted with [ID\\_MMFR0](#), [ID\\_MMFR2](#), [ID\\_MMFR3](#), 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\_MMFR1 is architecturally mapped to AArch64 System register [ID\\_MMFR1\\_EL1](#).

## Attributes

ID\_MMFR1 is a 32-bit register.

## Field descriptions

The ID\_MMFR1 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
<a href="#">BPred</a>				<a href="#">L1TstCln</a>				<a href="#">L1Uni</a>				<a href="#">L1Hvd</a>				<a href="#">L1UniSW</a>				<a href="#">L1HvdSW</a>				<a href="#">L1UniVA</a>				<a href="#">L1HvdVA</a>			

### BPred, bits [31:28]

Branch Predictor. Indicates branch predictor management requirements. Defined values are:

BPred	Meaning
0000	No branch predictor, or no MMU present. Implies a fixed MPU configuration.
0001	Branch predictor requires flushing on: <ul style="list-style-type: none"> <li>Enabling or disabling a stage of address translation.</li> <li>Writing new data to instruction locations.</li> <li>Writing new mappings to the translation tables.</li> <li>Changes to the <a href="#">TTBR0</a>, <a href="#">TTBR1</a>, or <a href="#">TTBCR</a> registers.</li> <li>Changes to the ContextID or ASID, or to the FCSE ProcessID if this is supported.</li> </ul>
0010	Branch predictor requires flushing on: <ul style="list-style-type: none"> <li>Enabling or disabling a stage of address translation.</li> <li>Writing new data to instruction locations.</li> <li>Writing new mappings to the translation tables.</li> <li>Any change to the <a href="#">TTBR0</a>, <a href="#">TTBR1</a>, or <a href="#">TTBCR</a> registers without a change to the corresponding ContextID or ASID, or FCSE ProcessID if this is supported.</li> </ul>
0011	Branch predictor requires flushing only on writing new data to instruction locations.
0100	For execution correctness, branch predictor requires no flushing at any time.

All other values are reserved.

In ARMv8-A the permitted values are 0010, 0011, or 0100. For values other than 0000 and 0100 the ARM Architecture Reference Manual, or the product documentation, might give more information about the required maintenance.

**L1TstCln, bits [27:24]**

Level 1 cache Test and Clean. Indicates the supported Level 1 data cache test and clean operations, for Harvard or unified cache implementations. Defined values are:

<b>L1TstCln</b>	<b>Meaning</b>
0000	None supported.
0001	Supported Level 1 data cache test and clean operations are: <ul style="list-style-type: none"> <li>• Test and clean data cache.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Test, clean, and invalidate data cache.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

**L1Uni, bits [23:20]**

Level 1 Unified cache. Indicates the supported entire Level 1 cache maintenance operations for a unified cache implementation. Defined values are:

<b>L1Uni</b>	<b>Meaning</b>
0000	None supported.
0001	Supported entire Level 1 cache operations are: <ul style="list-style-type: none"> <li>• Invalidate cache, including branch predictor if appropriate.</li> <li>• Invalidate branch predictor, if appropriate.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Clean cache, using a recursive model that uses the cache dirty status bit.</li> <li>• Clean and invalidate cache, using a recursive model that uses the cache dirty status bit.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

**L1Hvd, bits [19:16]**

Level 1 Harvard cache. Indicates the supported entire Level 1 cache maintenance operations for a Harvard cache implementation. Defined values are:

<b>L1Hvd</b>	<b>Meaning</b>
0000	None supported.
0001	Supported entire Level 1 cache operations are: <ul style="list-style-type: none"> <li>• Invalidate instruction cache, including branch predictor if appropriate.</li> <li>• Invalidate branch predictor, if appropriate.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Invalidate data cache.</li> <li>• Invalidate data cache and instruction cache, including branch predictor if appropriate.</li> </ul>
0011	As for 0010, and adds: <ul style="list-style-type: none"> <li>• Clean data cache, using a recursive model that uses the cache dirty status bit.</li> <li>• Clean and invalidate data cache, using a recursive model that uses the cache dirty status bit.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

**L1UniSW, bits [15:12]**

Level 1 Unified cache by Set/Way. Indicates the supported Level 1 cache line maintenance operations by set/way, for a unified cache implementation. Defined values are:

<b>L1UniSW</b>	<b>Meaning</b>
0000	None supported.
0001	Supported Level 1 unified cache line maintenance operations by set/way are: <ul style="list-style-type: none"> <li>• Clean cache line by set/way.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Clean and invalidate cache line by set/way.</li> </ul>
0011	As for 0010, and adds: <ul style="list-style-type: none"> <li>• Invalidate cache line by set/way.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

#### **L1HvdSW, bits [11:8]**

Level 1 Harvard cache by Set/Way. Indicates the supported Level 1 cache line maintenance operations by set/way, for a Harvard cache implementation. Defined values are:

<b>L1HvdSW</b>	<b>Meaning</b>
0000	None supported.
0001	Supported Level 1 Harvard cache line maintenance operations by set/way are: <ul style="list-style-type: none"> <li>• Clean data cache line by set/way.</li> <li>• Clean and invalidate data cache line by set/way.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Invalidate data cache line by set/way.</li> </ul>
0011	As for 0010, and adds: <ul style="list-style-type: none"> <li>• Invalidate instruction cache line by set/way.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

#### **L1UniVA, bits [7:4]**

Level 1 Unified cache by Virtual Address. Indicates the supported Level 1 cache line maintenance operations by VA, for a unified cache implementation. Defined values are:

<b>L1UniVA</b>	<b>Meaning</b>
0000	None supported.
0001	Supported Level 1 unified cache line maintenance operations by VA are: <ul style="list-style-type: none"> <li>• Clean cache line by VA.</li> <li>• Invalidate cache line by VA.</li> <li>• Clean and invalidate cache line by VA.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Invalidate branch predictor by VA, if branch predictor is implemented.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

#### **L1HvdVA, bits [3:0]**

Level 1 Harvard cache by Virtual Address. Indicates the supported Level 1 cache line maintenance operations by VA, for a Harvard cache implementation. Defined values are:

<b>L1HvdVA</b>	<b>Meaning</b>
0000	None supported.
0001	Supported Level 1 Harvard cache line maintenance operations by VA are: <ul style="list-style-type: none"> <li>• Clean data cache line by VA.</li> <li>• Invalidate data cache line by VA.</li> <li>• Clean and invalidate data cache line by VA.</li> <li>• Clean instruction cache line by VA.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Invalidate branch predictor by VA, if branch predictor is implemented.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

## Accessing the ID\_MMFR1

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, 5	000	101	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<sup>®</sup> 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.

# ID\_MMFR2, Memory Model Feature Register 2

The ID\_MMFR2 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\\_MMFR3](#), 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\_MMFR2 is architecturally mapped to AArch64 System register [ID\\_MMFR2\\_EL1](#).

## Attributes

ID\_MMFR2 is a 32-bit register.

## Field descriptions

The ID\_MMFR2 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
<a href="#">HWAaccFlg</a>				<a href="#">WFISall</a>				<a href="#">MemBarr</a>				<a href="#">UniTLB</a>				<a href="#">HvdTLB</a>				<a href="#">L1HvdRng</a>				<a href="#">L1HvdBG</a>				<a href="#">L1HvdFG</a>			

### HWAaccFlg, bits [31:28]

Hardware Access Flag. In earlier versions of the ARM Architecture, this field indicates support for a Hardware Access flag, as part of the VMSAv7 implementation. Defined values are:

HWAaccFlg	Meaning
0000	Not supported.
0001	Support for VMSAv7 Access flag, updated in hardware.

All other values are reserved.

In ARMv8 the only permitted value is 0000.

### WFISall, bits [27:24]

Wait For Interrupt Stall. Indicates the support for Wait For Interrupt (WFI) stalling. Defined values are:

WFISall	Meaning
0000	Not supported.
0001	Support for WFI stalling.

All other values are reserved.

In ARMv8 the permitted values are 0000 and 0001.

**MemBarr, bits [23:20]**

Memory Barrier. Indicates the supported memory barrier System instructions in the (coproc==1111) encoding space:

MemBarr	Meaning
0000	None supported.
0001	Supported memory barrier System instructions are: <ul style="list-style-type: none"> <li>• Data Synchronization Barrier (DSB).</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Instruction Synchronization Barrier (ISB).</li> <li>• Data Memory Barrier (DMB).</li> </ul>

All other values are reserved.

In ARMv8 the only permitted value is 0010.

ARM deprecates the use of these operations. ID\_ISAR4.Barrier\_instrs indicates the level of support for the preferred barrier instructions.

**UniTLB, bits [19:16]**

Unified TLB. Indicates the supported TLB maintenance operations, for a unified TLB implementation. Defined values are:

UniTLB	Meaning
0000	Not supported.
0001	Supported unified TLB maintenance operations are: <ul style="list-style-type: none"> <li>• Invalidate all entries in the TLB.</li> <li>• Invalidate TLB entry by VA.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Invalidate TLB entries by ASID match.</li> </ul>
0011	As for 0010, and adds: <ul style="list-style-type: none"> <li>• Invalidate instruction TLB and data TLB entries by VA All ASID. This is a shared unified TLB operation.</li> </ul>
0100	As for 0011, and adds: <ul style="list-style-type: none"> <li>• Invalidate Hyp mode unified TLB entry by VA.</li> <li>• Invalidate entire Non-secure PL1&amp;0 unified TLB.</li> <li>• Invalidate entire Hyp mode unified TLB.</li> </ul>
0101	As for 0100, and adds the following operations: <a href="#">TLBIMVALIS</a> , <a href="#">TLBIMVAALIS</a> , <a href="#">TLBIMVALHIS</a> , <a href="#">TLBIMVAL</a> , <a href="#">TLBIMVAAL</a> , <a href="#">TLBIMVALH</a> .
0110	As for 0101, and adds the following operations: <a href="#">TLBIIPAS2IS</a> , <a href="#">TLBIIPAS2LIS</a> , <a href="#">TLBIIPAS2</a> , <a href="#">TLBIIPAS2L</a> .

All other values are reserved.

In ARMv8-A the only permitted value is 0110.

**HvdTLB, bits [15:12]**

If the Unified TLB field (UniTLB, bits [19:16]) is not 0000, then the meaning of this field is IMPLEMENTATION DEFINED. ARM deprecates the use of this field by software.

**L1HvdRng, bits [11:8]**

Level 1 Harvard cache Range. Indicates the supported Level 1 cache maintenance range operations, for a Harvard cache implementation. Defined values are:

L1HvdRng	Meaning
0000	Not supported.
0001	Supported Level 1 Harvard cache maintenance range operations are: <ul style="list-style-type: none"> <li>• Invalidate data cache range by VA.</li> <li>• Invalidate instruction cache range by VA.</li> <li>• Clean data cache range by VA.</li> <li>• Clean and invalidate data cache range by VA.</li> </ul>

All other values are reserved.

In ARMv8 the only permitted value is 0000.

**L1HvdBG, bits [7:4]**

Level 1 Harvard cache Background fetch. Indicates the supported Level 1 cache background fetch operations, for a Harvard cache implementation. When supported, background fetch operations are non-blocking operations. Defined values are:

L1HvdBG	Meaning
0000	Not supported.
0001	Supported Level 1 Harvard cache background fetch operations are: <ul style="list-style-type: none"> <li>Fetch instruction cache range by VA.</li> <li>Fetch data cache range by VA.</li> </ul>

All other values are reserved.

In ARMv8 the only permitted value is 0000.

**L1HvdFG, bits [3:0]**

Level 1 Harvard cache Foreground fetch. Indicates the supported Level 1 cache foreground fetch operations, for a Harvard cache implementation. When supported, foreground fetch operations are blocking operations. Defined values are:

L1HvdFG	Meaning
0000	Not supported.
0001	Supported Level 1 Harvard cache foreground fetch operations are: <ul style="list-style-type: none"> <li>Fetch instruction cache range by VA.</li> <li>Fetch data cache range by VA.</li> </ul>

All other values are reserved.

In ARMv8 the only permitted value is 0000.

**Accessing the ID\_MMFR2**

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, 6	000	110	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<sup>®</sup> 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.

02/05/2017 15:43

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

# 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
<a href="#">Supersec</a>				<a href="#">CMemSz</a>				<a href="#">CohWalk</a>				<a href="#">PAN</a>				<a href="#">MaintBcst</a>				<a href="#">BPMaint</a>				<a href="#">CMaintSW</a>				<a href="#">CMaintVA</a>			

### 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 ARMv8.2 and ARMv8.1:

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

PAN	Meaning
0000	PAN not supported.
0001	PAN supported.
0010	PAN supported and <a href="#">ATS1CPRP</a> and <a href="#">ATS1CPWP</a> instructions supported.

All other values are reserved.

In ARMv8.0 the only permitted value is 0000.

In ARMv8.1 the only permitted value is 0001. This feature is identified by the name ARMv8.1-PAN.

From ARMv8.2, the only permitted value is 0010. This feature is identified by the name ARMv8.2-ATS1E1.

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"> <li>• Invalidate all branch predictors.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Invalidate branch predictors by VA.</li> </ul>

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"> <li>• Invalidate data cache by set/way.</li> <li>• Clean data cache by set/way.</li> <li>• Clean and invalidate data cache by set/way.</li> </ul>

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"> <li>• Invalidate data cache by VA.</li> <li>• Clean data cache by VA.</li> <li>• Clean and invalidate data cache by VA.</li> <li>• Invalidate instruction cache by VA.</li> <li>• Invalidate all instruction cache entries.</li> </ul>

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.

02/05/2017 15:43

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

# ID\_MMFR4, Memory Model Feature Register 4

The ID\_MMFR4 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\\_MMFR3](#).

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\_MMFR4 is architecturally mapped to AArch64 System register [ID\\_MMFR4\\_EL1](#).

In an implementation that does not include [ACTLR2](#) and [HACTLR2](#) this register is RAZ.

## Attributes

ID\_MMFR4 is a 32-bit register.

## Field descriptions

The ID\_MMFR4 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		LSM				HPDS				CnP				XNX				AC2				SpecSEI		

### Bits [31:24]

Reserved, RAZ.

### LSM, bits [23:20]

In ARMv8.2:

Indicates support for LSMAOE and nTLSMD bits in [HSCTLR](#) and [SCTLR](#). Defined values are:

LSM	Meaning
0000	LSMAOE and nTLSMD bits not supported.
0001	LSMAOE and nTLSMD bits supported.

All other values are reserved.

In ARMv8.0 and ARMv8.1 the only permitted value is 0000.

From ARMv8.2, the permitted values are 0000 and 0001. This feature is identified by the name ARMv8.2-LSMAOC.

### In ARMv8.1 and ARMv8.0:

Reserved, RAZ.

**HPDS, bits [19:16]****In ARMv8.2:**

Hierarchical permission disables bits in translation tables. Defined values are:

HPDS	Meaning
0000	Disabling of hierarchical controls not supported.
0001	Supports disabling of hierarchical controls using the <a href="#">TTBCR2</a> .HPD0, <a href="#">TTBCR2</a> .HPD1, and <a href="#">HTCR</a> .HPD bits.
0010	Supports disabling of hierarchical controls using the <a href="#">TTBCR2</a> .HPD0, <a href="#">TTBCR2</a> .HPD1, and <a href="#">HTCR</a> .HPD bits, and hardware allocation of bits[62:59] of the last level page table descriptor for IMPLEMENTATION DEFINED use.

All other values are reserved.

In ARMv8.0 and ARMv8.1 the only permitted value is 0000.

From ARMv8.2, the permitted values are 0000, 0001 and 0010. This feature is identified by the name ARMv8.2-AA32HPD.

**Note**

The encoding 0000 implies that the encoding for [TTBCR2](#) is unallocated.

**In ARMv8.1 and ARMv8.0:**

Reserved, RAZ.

**CnP, bits [15:12]****In ARMv8.2:**

Common not Private translations. Defined values are:

CnP	Meaning
0000	Common not Private translations not supported.
0001	Common not Private translations supported.

All other values are reserved.

In ARMv8.0 and ARMv8.1 the only permitted value is 0000.

From ARMv8.2, the only permitted value is 0001. This feature is identified by the name ARMv8.2-TTCNP.

**In ARMv8.1 and ARMv8.0:**

Reserved, RAZ.

**XNX, bits [11:8]****In ARMv8.2:**

Support for execute never control distinction at stage 2 bit. Defined values are:

XNX	Meaning
0000	Distinction between EL0 and EL1 execute permission at stage 2 not supported.
0001	Distinction between EL0 and EL1 execute permission at stage 2 supported.

All other values are reserved.

In ARMv8.0 and ARMv8.1 the only permitted value is 0000.

From ARMv8.2, the only permitted value is 0001. This feature is identified by the name ARMv8.2-TTS2UXN.

In ARMv8.1 and ARMv8.0:

Reserved, RAZ.

### AC2, bits [7:4]

Indicates the extension of the [ACTLR](#) and [HACTLR](#) registers using [ACTLR2](#) and [HACTLR2](#). Defined values are:

AC2	Meaning
0000	<a href="#">ACTLR2</a> and <a href="#">HACTLR2</a> are not implemented.
0001	<a href="#">ACTLR2</a> and <a href="#">HACTLR2</a> are implemented.

All other values are reserved.

In ARMv8.0 and ARMv8.1 the permitted values are 0000 and 0001.

From ARMv8.2, the only permitted value is 0001.

### SpecSEI, bits [3:0]

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

SpecSEI	Meaning
0000	The PE never generates an SError interrupt due to an external abort on a speculative read.
0001	The PE might generate an SError interrupt due to an external abort on a speculative read.

All other values are reserved.

When the RAS Extension is not implemented, this field is RAZ.

## Accessing the ID\_MMFR4

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, c2, 6	000	110	0000	1111	0010

## 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<sup>®</sup> 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 using AArch64.
- 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 using AArch64.
- 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.

02/05/2017 15:43

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

# ID\_PFR0, Processor Feature Register 0

The ID\_PFR0 characteristics are:

## Purpose

Gives top-level information about the instruction sets supported by the PE in AArch32 state.

Must be interpreted with [ID\\_PFR1](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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\_PFR0 is architecturally mapped to AArch64 System register [ID\\_PFR0\\_EL1](#).

## Attributes

ID\_PFR0 is a 32-bit register.

## Field descriptions

The ID\_PFR0 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
RAS				0	0	0	0	0	0	0	0	0	0	0	0	State3				State2				State1				State0			

### 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.

### Bits [27:16]

Reserved, RES0.

### State3, bits [15:12]

T32EE instruction set support. Defined values are:

State3	Meaning
0000	Not implemented.
0001	T32EE instruction set implemented.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

**State2, bits [11:8]**

Jazelle extension support. Defined values are:

State2	Meaning
0000	Not implemented.
0001	Jazelle extension implemented, without clearing of <a href="#">JOSCR</a> .CV on exception entry.
0010	Jazelle extension implemented, with clearing of <a href="#">JOSCR</a> .CV on exception entry.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**State1, bits [7:4]**

T32 instruction set support. Defined values are:

State1	Meaning
0000	T32 instruction set not implemented.
0001	T32 encodings before the introduction of Thumb-2 technology implemented: <ul style="list-style-type: none"> <li>• All instructions are 16-bit.</li> <li>• A BL or BLX is a pair of 16-bit instructions.</li> <li>• 32-bit instructions other than BL and BLX cannot be encoded.</li> </ul>
0011	T32 encodings after the introduction of Thumb-2 technology implemented, for all 16-bit and 32-bit T32 basic instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0011.

**State0, bits [3:0]**

A32 instruction set support. Defined values are:

State0	Meaning
0000	A32 instruction set not implemented.
0001	A32 instruction set implemented.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**Accessing the ID\_PFR0**

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, 0	000	000	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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  :

- 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.

02/05/2017 15:43

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

# ID\_PFR1, Processor Feature Register 1

The ID\_PFR1 characteristics are:

## Purpose

Gives information about the AArch32 programmers' model.

Must be interpreted with [ID\\_PFR0](#).

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\_PFR1 is architecturally mapped to AArch64 System register [ID\\_PFR1\\_EL1](#).

## Attributes

ID\_PFR1 is a 32-bit register.

## Field descriptions

The ID\_PFR1 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
GIC				Virt_frac				Sec_frac				GenTimer				Virtualization				MProgMod				Security				ProgMod			

### GIC, bits [31:28]

System register GIC CPU interface. Defined values are:

GIC	Meaning
0000	No System register interface to the GIC CPU interface 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.

### Virt\_frac, bits [27:24]

Virtualization fractional field. When the Virtualization field is 0000, determines the support for features from the ARMv7 Virtualization Extensions. Defined values are:

Virt_frac	Meaning
0000	No features from the ARMv7 Virtualization Extensions are implemented.
0001	The following features of the ARMv7 Virtualization Extensions are implemented: <ul style="list-style-type: none"> <li>The <a href="#">SCR</a>.SIF bit, if EL3 is implemented.</li> <li>The modifications to the <a href="#">SCR</a>.AW and <a href="#">SCR</a>.FW bits described in the Virtualization Extensions, if EL3 is implemented.</li> <li>The MSR (Banked register) and MRS (Banked register) instructions.</li> <li>The ERET instruction.</li> </ul>

All other values are reserved.

In ARMv8-A the permitted values are:

- 0000 when EL2 is implemented.
- 0001 when EL2 is not implemented.

This field is only valid when the value of ID\_PFR1.Virtualization is 0, otherwise it holds the value 0000.

---

#### Note

The ID\_ISAR registers do not identify whether the instructions added by the ARMv7 Virtualization Extensions are implemented.

---

### Sec\_frac, bits [23:20]

Security fractional field. When the Security field is 0000, determines the support for features from the ARMv7 Security Extensions. Defined values are:

Sec_frac	Meaning
0000	No features from the ARMv7 Security Extensions are implemented.
0001	The following features from the ARMv7 Security Extensions are implemented: <ul style="list-style-type: none"> <li>• The VBAR register.</li> <li>• The <a href="#">TTBCR</a>.PD0 and <a href="#">TTBCR</a>.PD1 bits.</li> </ul>
0010	As for 0001, plus the ability to access Secure or Non-secure physical memory is supported.

---

All other values are reserved.

In ARMv8-A the permitted values are:

- 0000 when EL3 is implemented.
- 0001 or 0010 when EL3 is not implemented.

This field is only valid when the value of ID\_PFR1.Security is 0, otherwise it holds the value 0000.

### GenTimer, bits [19:16]

Generic Timer support. Defined values are:

GenTimer	Meaning
0000	Not implemented.
0001	Generic Timer implemented.

---

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Virtualization, bits [15:12]

Virtualization support. Defined values are:

Virtualization	Meaning
0000	EL2, Hyp mode, and the HVC instruction not implemented.
0001	EL2, Hyp mode, the HVC instruction, and all the features described by Virt_frac == 0001 implemented.

---

All other values are reserved.

In ARMv8-A the permitted values are:

- 0000 when EL2 is not implemented.
- 0001 when EL2 is implemented.

In an implementation that includes EL2, if EL2 cannot use AArch32 but EL1 can use AArch32 then this field has the value 0001.

---

#### Note

---

The ID\_ISARs do not identify whether the HVC instruction is implemented.

### MProgMod, bits [11:8]

M profile programmers' model support. Defined values are:

MProgMod	Meaning
0000	Not supported.
0010	Support for two-stack programmers' model.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### Security, bits [7:4]

Security support. Defined values are:

Security	Meaning
0000	EL3, Monitor mode, and the SMC instruction not implemented.
0001	EL3, Monitor mode, the SMC instruction, and all the features described by Sec_frac == 0001 implemented.
0010	As for 0001, and adds the ability to set the <a href="#">NSACR</a> .RFR bit. Not permitted in ARMv8 as the <a href="#">NSACR</a> .RFR bit is RES0.

All other values are reserved.

In ARMv8-A the permitted values are:

- 0000 when EL3 is not implemented.
- 0001 when EL3 is implemented.

In an implementation that includes EL3, if EL3 cannot use AArch32 but EL1 can use AArch32 then this field has the value 0001.

### ProgMod, bits [3:0]

Support for the standard programmers' model for ARMv4 and later. Model must support User, FIQ, IRQ, Supervisor, Abort, Undefined, and System modes. Defined values are:

ProgMod	Meaning
0000	Not supported.
0001	Supported.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

## Accessing the ID\_PFR1

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, 1	000	001	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<sup>®</sup> 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.



# IFAR, Instruction Fault Address Register

The IFAR characteristics are:

## Purpose

Holds the virtual address of the faulting address that caused a synchronous Prefetch Abort exception.

This register is part of the Exception and fault handling registers functional group.

## Configuration

AArch32 System register IFAR (NS) is architecturally mapped to AArch64 System register [FAR\\_EL1\[63:32\]](#).

AArch32 System register IFAR (S) is architecturally mapped to AArch32 System register [HIFAR](#) when EL2 is implemented.

AArch32 System register IFAR (S) is architecturally mapped to AArch64 System register [FAR\\_EL2\[63:32\]](#) when EL2 is implemented.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

IFAR is a 32-bit register.

## Field descriptions

The IFAR 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 of faulting address of synchronous Prefetch Abort exception																															

### Bits [31:0]

VA of faulting address of synchronous Prefetch Abort exception.

## Accessing the IFAR

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>, c6, c0, 2	000	010	0110	1111	0000

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	

EL3 using AArch32	x	x	0	-	n/a	n/a	RW	IFAR_s
EL3 not implemented	x	x	0	-	RW	n/a	n/a	IFAR
EL3 not implemented	x	0	1	-	RW	RW	n/a	IFAR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	IFAR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	IFAR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	IFAR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	IFAR
EL3 using AArch32	x	0	1	-	RW	RW	RW	IFAR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	IFAR_ns

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<sup>®</sup> 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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.T6](#)=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.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.T6](#)=1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- 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.T6](#)=1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# IFSR, Instruction Fault Status Register

The IFSR characteristics are:

## Purpose

Holds status information about the last instruction fault.

This register is part of the Exception and fault handling registers functional group.

## Configuration

AArch32 System register IFSR is architecturally mapped to AArch64 System register [IFSR32\\_EL2](#).

The current translation table format determines which format of the register is used.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

IFSR is a 32-bit register.

## Field descriptions

The IFSR bit assignments are:

### When TTBCR.EAE==0:

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

### Bits [31:17]

Reserved, RES0.

### FnV, bit [16]

FAR not Valid, for a Synchronous external abort other than a Synchronous external abort on a translation table walk.

FnV	Meaning
0	<a href="#">IFAR</a> is valid.
1	<a href="#">IFAR</a> is not valid, and holds an UNKNOWN value.

This field is only valid for a Synchronous external abort other than a Synchronous external abort on a translation table walk. It is RES0 for all other Prefetch Abort exceptions.

### Bits [15:13]

Reserved, RES0.

### ExT, bit [12]

External abort type. This bit can be used to provide an IMPLEMENTATION DEFINED classification of external aborts.

In an implementation that does not provide any classification of external aborts, this bit is RES0.

For aborts other than external aborts this bit always returns 0.

### Bit [11]

Reserved, RES0.

### FS[4], bit [10]

See FS[3:0], bits [3:0] for description of the FS field.

### LPAE, bit [9]

On taking a Data Abort exception, this bit is set as follows:

LPAE	Meaning
0	Using the Short-descriptor translation table formats.
1	Using the Long-descriptor translation table formats.

Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.

### Bits [8:4]

Reserved, RES0.

### FS[3:0], bits [3:0]

Fault status bits. Interpreted with bit [10]. Possible values of FS[4:0] are:

FS	Meaning
00001	PC alignment fault
00010	Debug exception
00011	Access flag fault, level 1
00101	Translation fault, level 1
00110	Access flag fault, level 2
00111	Translation fault, level 2
01000	Synchronous external abort, not on translation table walk
01001	Domain fault, level 1
01011	Domain fault, level 2
01100	Synchronous external abort, on translation table walk, level 1
01101	Permission fault, level 1
01110	Synchronous external abort, on translation table walk, level 2
01111	Permission fault, level 2
10000	TLB conflict abort
10100	IMPLEMENTATION DEFINED fault (Lockdown fault)
11001	Synchronous parity or ECC error on memory access, not on translation table walk
11100	Synchronous parity or ECC error on translation table walk, level 1
11110	Synchronous parity or ECC error on translation table walk, level 2

All other values are reserved.

When the RAS Extension is implemented, 11001, 11100, and 11110, are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults on a Short-descriptor translation table lookup' in the ARMv8 ARM.

### When TTBCR.EAE==1:

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

**Bits [31:17]**

Reserved, RES0.

**FnV, bit [16]**

FAR not Valid, for a Synchronous external abort other than a Synchronous external abort on a translation table walk.

FnV	Meaning
0	<a href="#">IFAR</a> is valid.
1	<a href="#">IFAR</a> is not valid, and holds an UNKNOWN value.

This field is only valid for a Synchronous external abort other than a Synchronous external abort on a translation table walk. It is RES0 for all other Prefetch Abort exceptions.

**Bits [15:13]**

Reserved, RES0.

**ExT, bit [12]**

External abort type. This bit can be used to provide an IMPLEMENTATION DEFINED classification of external aborts.

In an implementation that does not provide any classification of external aborts, this bit is RES0.

For aborts other than external aborts this bit always returns 0.

**Bits [11:10]**

Reserved, RES0.

**LPAE, bit [9]**

On taking a Data Abort exception, this bit is set as follows:

LPAE	Meaning
0	Using the Short-descriptor translation table formats.
1	Using the Long-descriptor translation table formats.

Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.

**Bits [8:6]**

Reserved, RES0.

**STATUS, bits [5:0]**

Fault status bits. Possible values of this field are:

STATUS	Meaning
000000	Address size fault in <a href="#">TTBR0</a> or <a href="#">TTBR1</a>
000001	Address size fault, level 1
000010	Address size fault, level 2
000011	Address size fault, level 3
000101	Translation fault, level 1
000110	Translation fault, level 2
000111	Translation fault, level 3
001001	Access flag fault, level 1
001010	Access flag fault, level 2
001011	Access flag fault, level 3
001101	Permission fault, level 1
001110	Permission fault, level 2
001111	Permission fault, level 3
010000	Synchronous external abort, not on translation table walk
010101	Synchronous external abort, on translation table walk, level 1
010110	Synchronous external abort, on translation table walk, level 2
010111	Synchronous external abort, on translation table walk, level 3
011000	Synchronous parity or ECC error on memory access, not on translation table walk
011101	Synchronous parity or ECC error on memory access on translation table walk, level 1
011110	Synchronous parity or ECC error on memory access on translation table walk, level 2
011111	Synchronous parity or ECC error on memory access on translation table walk, level 3
100001	PC alignment fault
100010	Debug exception
110000	TLB conflict abort

All other values are reserved.

When the RAS Extension is implemented, 011000, 011101, 011110, and 011111, are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults on a Long-descriptor translation table lookup' in the ARMv8 ARM.

## Accessing the IFSR

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>, c5, c0, 1	000	001	0101	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	IFSR
EL3 not implemented	x	0	1	-	RW	RW	n/a	IFSR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	IFSR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	IFSR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	IFSR

EL3 using AArch64	x	1	1	-	n/a	RW	n/a	IFSR
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	IFSR_s
EL3 using AArch32	x	0	1	-	RW	RW	RW	IFSR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	IFSR_ns

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.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.T5==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.T5==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.T5==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.

# ISR, Interrupt Status Register

The ISR characteristics are:

## Purpose

Shows whether any IRQ, FIQ, or external abort is pending. In an implementation that includes EL2, when the register is accessed from Non-secure EL1, a pending interrupt might be a physical interrupt or a virtual interrupt, and the architecture does not provide any mechanism that software executing at Non-secure EL1 can use to determine whether a pending interrupt is physical or virtual. For all other accesses, any indicated interrupt must be a physical interrupt.

This register is part of the Exception and fault handling registers functional group.

## Configuration

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

AArch32 System register ISR is architecturally mapped to AArch64 System register [ISR\\_EL1](#).

## Attributes

ISR is a 32-bit register.

## Field descriptions

The ISR 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	A	I	F	0	0	0	0	0	0

### Bits [31:9]

Reserved, RES0.

### A, bit [8]

Asynchronous external abort pending bit:

A	Meaning
0	No pending asynchronous external abort.
1	An asynchronous external abort is pending.

### I, bit [7]

IRQ pending bit. Indicates whether an IRQ interrupt is pending:

I	Meaning
0	No pending IRQ.
1	An IRQ interrupt is pending.

### F, bit [6]

FIQ pending bit. Indicates whether an FIQ interrupt is pending.

F	Meaning
0	No pending FIQ.
1	An FIQ interrupt is pending.



**Bits [5:0]**

Reserved, RES0.

**Accessing the ISR**

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>, c12, c1, 0	000	000	1100	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<sup>®</sup> 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.T12==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.T12==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.T12==1](#), Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

# JIDR, Jazelle ID Register

The JIDR characteristics are:

## Purpose

A Jazelle register, which identified the Jazelle architecture version.

This register is part of the Legacy feature registers functional group.

## Configuration

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

## Attributes

JIDR is a 32-bit register.

## Field descriptions

The JIDR 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	0

### Bits [31:0]

RO, RAZ at EL1, EL2, and EL3. It is IMPLEMENTATION DEFINED whether this field is RAZ or UNDEFINED at EL0.

## Accessing the JIDR

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
p14, 7, <Rt>, c0, c0, 0	111	000	0000	1110	0000

## Accessibility

The register is accessible as follows:

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

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

For accesses from EL0 it is IMPLEMENTATION DEFINED whether the register is RO or UNDEFINED.

## 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HCR\\_EL2](#).TID0==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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- If [HCR\\_EL2](#).TID0==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR](#).TID0==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# JMCR, Jazelle Main Configuration Register

The JMCR characteristics are:

## Purpose

A Jazelle register, which provides control of the Jazelle extension.

This register is part of the Legacy feature registers functional group.

## Configuration

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

## Attributes

JMCR is a 32-bit register.

## Field descriptions

The JMCR 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	0

### Bits [31:0]

RAZ/WI at EL1, EL2, and EL3. It is IMPLEMENTATION DEFINED whether this field is RAZ/WI or UNDEFINED at EL0.

## Accessing the JMCR

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, 7, <Rt>, c2, c0, 0	111	000	0010	1110	0000

## Accessibility

The register is accessible as follows:

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

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

For accesses from EL0 it is IMPLEMENTATION DEFINED whether the register is RW or UNDEFINED.

02/05/2017 15:43

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

# JOSCR, Jazelle OS Control Register

The JOSCR characteristics are:

## Purpose

A Jazelle register, which provides operating system control of the Jazelle Extension.

This register is part of the Legacy feature registers functional group.

## Configuration

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

## Attributes

JOSCR is a 32-bit register.

## Field descriptions

The JOSCR 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	0

### Bits [31:0]

RAZ/WI at EL1, EL2, and EL3. It is IMPLEMENTATION DEFINED whether this field is RAZ/WI or UNDEFINED at EL0.

## Accessing the JOSCR

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, 7, <Rt>, c1, c0, 0	111	000	0001	1110	0000

## Accessibility

The register is accessible as follows:

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

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

For accesses from EL0 it is IMPLEMENTATION DEFINED whether the register is RW or UNDEFINED.

02/05/2017 15:43

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

# MAIR0, Memory Attribute Indirection Register 0

The MAIR0 characteristics are:

## Purpose

Along with [MAIR1](#), provides the memory attribute encodings corresponding to the possible AttrIdx values in a Long-descriptor format translation table entry for stage 1 translations.

AttrIdx[2] indicates the MAIR register to be used:

- When AttrIdx[2] is 0, MAIR0 is used.
- When AttrIdx[2] is 1, [MAIR1](#) is used.

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch32 System register MAIR0 is architecturally mapped to AArch64 System register [MAIR\\_EL1\[31:0\]](#) when TTBCR.EAE==1.

MAIR0 and [PRRR](#) are the same register, with a different view depending on the value of [TTBCR.EAE](#):

- When it is set to 0, the register is as described in [PRRR](#).
- When it is set to 1, the register is as described in MAIR0.

When EL3 is using AArch32, write access to MAIR0(S) is disabled when the CP15SDISABLE signal is asserted HIGH.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

MAIR0 is a 32-bit register.

## Field descriptions

The MAIR0 bit assignments are:

### When TTBCR.EAE==1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Attr3								Attr2								Attr1								Attr0							

### Attr<n>, bits [8n+7:8n], for n = 0 to 3

The memory attribute encoding for an AttrIdx[2:0] entry in a Long descriptor format translation table entry, where:

- AttrIdx[2:0] gives the value of <n> in Attr<n>.
- AttrIdx[2] defines which MAIR to access. Attr7 to Attr4 are in MAIR1, and Attr3 to Attr0 are in MAIR0.

Bits [7:4] are encoded as follows:



Attr<n>[7:4]	Meaning
0000	Device memory. See encoding of Attr<n>[3:0] for the type of Device memory.
00RW, RW not 00	Normal memory, Outer Write-Through Transient
0100	Normal memory, Outer Non-cacheable
01RW, RW not 00	Normal memory, Outer Write-Back Transient
10RW	Normal memory, Outer Write-Through Non-transient
11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

The meaning of bits [3:0] depends on the value of bits [7:4]:

Attr<n>[3:0]	Meaning when Attr<n>[7:4] is 0000	Meaning when Attr<n>[7:4] is not 0000
0000	Device-nGnRnE memory	UNPREDICTABLE
00RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Transient
0100	Device-nGnRE memory	Normal memory, Inner Non-cacheable
01RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Transient
1000	Device-nGRE memory	Normal memory, Inner Write-Through Non-transient (RW=00)
10RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Non-transient
1100	Device-GRE memory	Normal memory, Inner Write-Back Non-transient (RW=00)
11RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Non-transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in some Attr<n> fields have the following meanings:

R or W	Meaning
0	No Allocate
1	Allocate

## Accessing the MAIR0

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>, c10, c2, 0	000	000	1010	1111	0010

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	MAIR0_s
EL3 not implemented	x	x	0	-	RW	n/a	n/a	MAIR0
EL3 not implemented	x	0	1	-	RW	RW	n/a	MAIR0

EL3 not implemented	x	1	1	-	n/a	RW	n/a	MAIR0
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	MAIR0
EL3 using AArch64	x	0	1	-	RW	RW	n/a	MAIR0
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	MAIR0
EL3 using AArch32	x	0	1	-	RW	RW	RW	MAIR0_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	MAIR0_ns

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

When EL3 is using AArch32, write access to MAIR0\_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.T10==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.T10==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.T10==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# MAIR1, Memory Attribute Indirection Register 1

The MAIR1 characteristics are:

## Purpose

Along with [MAIR0](#), provides the memory attribute encodings corresponding to the possible AttrIdx values in a Long-descriptor format translation table entry for stage 1 translations.

AttrIdx[2] indicates the MAIR register to be used:

- When AttrIdx[2] is 0, [MAIR0](#) is used.
- When AttrIdx[2] is 1, MAIR1 is used.

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch32 System register MAIR1 is architecturally mapped to AArch64 System register [MAIR\\_EL1\[63:32\]](#) when TTBCR.EAE==1.

MAIR1 and [NMRR](#) are the same register, with a different view depending on the value of [TTBCR](#).EAE:

- When it is set to 0, the register is as described in [NMRR](#).
- When it is set to 1, the register is as described in MAIR1.

When EL3 is using AArch32, write access to MAIR1(S) is disabled when the CP15SDISABLE signal is asserted HIGH.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

MAIR1 is a 32-bit register.

## Field descriptions

The MAIR1 bit assignments are:

### When TTBCR.EAE==1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<a href="#">Attr7</a>								<a href="#">Attr6</a>								<a href="#">Attr5</a>								<a href="#">Attr4</a>							

**Attr<n>, bits [8(n-4)+7:8(n-4)], for n = 4 to 7**

The memory attribute encoding for an AttrIdx[2:0] entry in a Long descriptor format translation table entry, where:

- AttrIdx[2:0] gives the value of <n> in Attr<n>.
- AttrIdx[2] defines which MAIR to access. Attr7 to Attr4 are in MAIR1, and Attr3 to Attr0 are in MAIR0.

Bits [7:4] are encoded as follows:

Attr<n>[7:4]	Meaning
0000	Device memory. See encoding of Attr<n>[3:0] for the type of Device memory.
00RW, RW not 00	Normal memory, Outer Write-Through Transient
0100	Normal memory, Outer Non-cacheable
01RW, RW not 00	Normal memory, Outer Write-Back Transient
10RW	Normal memory, Outer Write-Through Non-transient
11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

The meaning of bits [3:0] depends on the value of bits [7:4]:

Attr<n>[3:0]	Meaning when Attr<n>[7:4] is 0000	Meaning when Attr<n>[7:4] is not 0000
0000	Device-nGnRnE memory	UNPREDICTABLE
00RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Transient
0100	Device-nGnRE memory	Normal memory, Inner Non-cacheable
01RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Transient
1000	Device-nGRE memory	Normal memory, Inner Write-Through Non-transient (RW=00)
10RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Non-transient
1100	Device-GRE memory	Normal memory, Inner Write-Back Non-transient (RW=00)
11RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Non-transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in some Attr<n> fields have the following meanings:

R or W	Meaning
0	No Allocate
1	Allocate

## Accessing the MAIR1

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>, c10, c2, 1	000	001	1010	1111	0010

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	MAIR1_s
EL3 using AArch32	x	0	1	-	RW	RW	RW	MAIR1_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	MAIR1_ns

EL3 not implemented	x	x	0	-	RW	n/a	n/a	MAIR1
EL3 not implemented	x	0	1	-	RW	RW	n/a	MAIR1
EL3 not implemented	x	1	1	-	n/a	RW	n/a	MAIR1
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	MAIR1
EL3 using AArch64	x	0	1	-	RW	RW	n/a	MAIR1
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	MAIR1

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

When EL3 is using AArch32, write access to MAIR1\_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.T10==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.T10==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.T10==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.

# 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 'Identification registers, functional group' in the ARMv8 ARM, section G4.18.1.

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<sup>®</sup> 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.

02/05/2017 15:43

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



# MPIDR, Multiprocessor Affinity Register

The MPIDR characteristics are:

## Purpose

In a multiprocessor system, provides an additional PE identification mechanism for scheduling purposes.

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 MPIDR is architecturally mapped to AArch64 System register [MPIDR\\_EL1](#).

The assigned value of the MPIDR.{Aff2, Aff1, Aff0} or [MPIDR\\_EL1](#).{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole.

In a uniprocessor system ARM recommends that each Aff<n> field of this register returns a value of 0.

## Attributes

MPIDR is a 32-bit register.

## Field descriptions

The MPIDR 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
M	U	0	0	0	0	0	MT	Aff2								Aff1								Aff0							

### M, bit [31]

Indicates whether this implementation includes the functionality introduced by the ARMv7 Multiprocessing Extensions. The possible values of this bit are:

M	Meaning
0	This implementation does not include the ARMv7 Multiprocessing Extensions functionality.
1	This implementation includes the ARMv7 Multiprocessing Extensions functionality.

In ARMv8 this bit is RES1.

### U, bit [30]

Indicates a Uniprocessor system, as distinct from PE 0 in a multiprocessor system. The possible values of this bit are:

U	Meaning
0	Processor is part of a multiprocessor system.
1	Processor is part of a uniprocessor system.

### Bits [29:25]

Reserved, RES0.

**MT, bit [24]**

Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. The possible values of this bit are:

MT	Meaning
0	Performance of PEs at the lowest affinity level is largely independent.
1	Performance of PEs at the lowest affinity level is very interdependent.

**Aff2, bits [23:16]**

Affinity level 2. The least significant affinity level field, for this PE in the system.

**Aff1, bits [15:8]**

Affinity level 1. The intermediate affinity level field, for this PE in the system.

**Aff0, bits [7:0]**

Affinity level 0. The most significant affinity level field, for this PE in the system.

## Accessing the MPIDR

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, 5	000	101	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<sup>®</sup> 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.

02/05/2017 15:43

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

# MVBAR, Monitor Vector Base Address Register

The MVBAR characteristics are:

## Purpose

When EL3 is implemented and can use AArch32, holds the vector base address for any exception that is taken to Monitor mode.

Secure software must program the MVBAR with the required initial value as part of the PE boot sequence.

This register is part of:

- The Exception and fault handling registers functional group.
- The Security registers functional group.

## Configuration

This register is only accessible in Secure state.

It is IMPLEMENTATION DEFINED whether MVBAR[0] has a fixed value and ignored writes, or takes the last value written to it.

Write access to MVBAR is disabled when the CP15SDISABLE signal is asserted HIGH.

On a reset into EL3 using AArch32, the reset value of MVBAR is an IMPLEMENTATION DEFINED choice between:

- MVBAR[31:5] = an IMPLEMENTATION DEFINED value, which might be UNKNOWN.
- MVBAR[4:1] = RES0.
- MVBAR[0] = 0.

And:

- MVBAR[31:1] = an IMPLEMENTATION DEFINED value that is bits[31:1] of the AArch32 reset address.
- MVBAR[0] = 1.

## Attributes

MVBAR is a 32-bit register.

## Field descriptions

The MVBAR bit assignments are:

### When programmed with a vector base address:

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

#### Bits [31:5]

Vector Base Address. Bits[31:5] of the base address of the exception vectors for exceptions taken to this Exception level. Bits[4:0] of an exception vector are the exception offset.

#### Reserved, bits [4:0]

Reserved, see Configurations.

## Accessing the MVBAR

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, c0, 1	000	001	1100	1111	0000

## 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.

When EL3 is using AArch32, write access to MVBAR is UNDEFINED when the CP15SDISABLE signal is asserted HIGH.

If EL3 is implemented and is using AArch64, any read or write to MVBAR 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](#).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.E2H==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.

# MVFR0, Media and VFP Feature Register 0

The MVFR0 characteristics are:

## Purpose

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with [MVFR1](#) and [MVFR2](#).

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 Floating-point registers functional group.
- 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 MVFR0 is architecturally mapped to AArch64 System register [MVFR0\\_EL1](#).

Implemented only if the implementation includes Advanced SIMD and floating-point instructions.

## Attributes

MVFR0 is a 32-bit register.

## Field descriptions

The MVFR0 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
<a href="#">FPRound</a>				<a href="#">FPShVec</a>				<a href="#">FPSqrt</a>				<a href="#">FPDivide</a>				<a href="#">FPTrap</a>				<a href="#">FPDP</a>				<a href="#">FPSP</a>				<a href="#">SIMDReg</a>			

### FPRound, bits [31:28]

Floating-Point Rounding modes. Indicates whether the floating-point implementation provides support for rounding modes. Defined values are:

FPRound	Meaning
0000	Not implemented, or only Round to Nearest mode supported, except that Round towards Zero mode is supported for VCVT instructions that always use that rounding mode regardless of the <a href="#">FPSCR</a> setting.
0001	All rounding modes supported.

All other values are reserved.

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

### FPShVec, bits [27:24]

Short Vectors. Indicates whether the floating-point implementation provides support for the use of short vectors. Defined values are:

FPShVec	Meaning
0000	Short vectors not supported.
0001	Short vector operation supported.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### FPSqrt, bits [23:20]

Square Root. Indicates whether the floating-point implementation provides support for the ARMv6 VFP square root operations. Defined values are:

FPSqrt	Meaning
0000	Not supported in hardware.
0001	Supported.

All other values are reserved.

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

The VSQRT.F32 instruction also requires the single-precision floating-point attribute, bits [7:4], and the VSQRT.F64 instruction also requires the double-precision floating-point attribute, bits [11:8].

### FPDivide, bits [19:16]

Indicates whether the floating-point implementation provides support for VFP divide operations. Defined values are:

FPDivide	Meaning
0000	Not supported in hardware.
0001	Supported.

All other values are reserved.

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

The VDIV.F32 instruction also requires the single-precision floating-point attribute, bits [7:4], and the VDIV.F64 instruction also requires the double-precision floating-point attribute, bits [11:8].

### FPTrap, bits [15:12]

Floating Point Exception Trapping. Indicates whether the floating-point implementation provides support for exception trapping. Defined values are:

FPTrap	Meaning
0000	Not supported.
0001	Supported.

All other values are reserved.

A value of 0001 indicates that, when the corresponding trap is enabled, a floating-point exception generates an exception.

### FPDP, bits [11:8]

Double Precision. Indicates whether the floating-point implementation provides support for double-precision operations. Defined values are:

FPDP	Meaning
0000	Not supported in hardware.
0001	Supported, VFPv2.
0010	Supported, VFPv3, VFPv4, or ARMv8. VFPv3 and ARMv8 add an instruction to load a double-precision floating-point constant, and conversions between double-precision and fixed-point values.

All other values are reserved.

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

A value of 0b0001 or 0b0010 indicates support for all VFP double-precision instructions in the supported version of VFP, except that, in addition to this field being nonzero:

- VSQRT.F64 is only available if the Square root field is 0001.
- VDIV.F64 is only available if the Divide field is 0001.
- Conversion between double-precision and single-precision is only available if the single-precision field is nonzero.

**FPSP, bits [7:4]**

Single Precision. Indicates whether the floating-point implementation provides support for single-precision operations. Defined values are:

FPSP	Meaning
0000	Not supported in hardware.
0001	Supported, VFPv2.
0010	Supported, VFPv3 or VFPv4. VFPv3 adds an instruction to load a single-precision floating-point constant, and conversions between single-precision and fixed-point values.

All other values are reserved.

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

A value of 0b0001 or 0b0010 indicates support for all VFP single-precision instructions in the supported version of VFP, except that, in addition to this field being nonzero:

- VSQRT.F32 is only available if the Square root field is 0001.
- VDIV.F32 is only available if the Divide field is 0001.
- Conversion between double-precision and single-precision is only available if the double-precision field is nonzero.

**SIMDReg, bits [3:0]**

Advanced SIMD registers. Indicates whether the Advanced SIMD and floating-point implementation provides support for the Advanced SIMD and floating-point register bank. Defined values are:

SIMDReg	Meaning
0000	The implementation has no Advanced SIMD and floating-point support.
0001	The implementation includes floating-point support with 16 x 64-bit registers.
0010	The implementation includes Advanced SIMD and floating-point support with 32 x 64-bit registers.

All other values are reserved.

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

**Accessing the MVFR0**

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

```
VMRS <Rt>, <spec_reg>
```

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

<spec_reg>	reg
MVFR0	0111

**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 `HCR_EL2.E2H==0` :

- If `CPACR.cp10==00`, read accesses to this register from PL1 are UNDEFINED.

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

- If `CPTR_EL2.TFP==1`, Non-secure read accesses to this register from EL1 are trapped to EL2.
- 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 `CPTR_EL2.FPEN==00`, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If `CPTR_EL2.FPEN==10`, Non-secure read accesses to this register from EL1 are trapped to EL2.
- 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 AArch32 and `SCR_EL3.NS==1` :

- If `HCPTR.TCP10==1`, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.
- If `HCPTR.TCP10==1`, Non-secure read accesses to this register from EL2 are UNDEFINED.
- If `HCR.TID3==1`, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

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

- If `NSACR.cp10==0`, Non-secure read accesses to this register from EL1 and EL2 are UNDEFINED.

When EL3 is implemented and is using AArch64 :

- If `CPTR_EL3.TFP==1`, read accesses to this register from EL1 and EL2 are trapped to EL3.

02/05/2017 15:43

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

# MVFR1, Media and VFP Feature Register 1

The MVFR1 characteristics are:

## Purpose

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with [MVFR0](#) and [MVFR2](#).

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 Floating-point registers functional group.
- 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 MVFR1 is architecturally mapped to AArch64 System register [MVFR1\\_EL1](#).

Implemented only if the implementation includes Advanced SIMD and floating-point instructions.

## Attributes

MVFR1 is a 32-bit register.

## Field descriptions

The MVFR1 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
<a href="#">SIMDFMAC</a>				<a href="#">FPHP</a>				<a href="#">SIMDHP</a>				<a href="#">SIMDSP</a>				<a href="#">SIMDInt</a>				<a href="#">SIMDLS</a>				<a href="#">FPDNaN</a>				<a href="#">FPFtZ</a>			

### SIMDFMAC, bits [31:28]

Advanced SIMD Fused Multiply-Accumulate. Indicates whether the Advanced SIMD implementation provides fused multiply accumulate instructions. Defined values are:

SIMDFMAC	Meaning
0000	Not implemented.
0001	Implemented.

All other values are reserved.

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

The Advanced SIMD and floating-point implementations must provide the same level of support for these instructions.

### FPHP, bits [27:24]

Floating Point Half Precision. Indicates the level of half-precision floating-point support. Defined values are:

<b>FPHP</b>	<b>Meaning</b>
0000	Not supported.
0001	Floating-point half-precision conversion instructions are supported for conversion between single-precision and half-precision.
0010	Floating-point half-precision conversion instructions are supported for conversion between single-precision and half-precision and between double-precision and half-precision.
0011	As for 0010, and also includes support for half-precision floating-point arithmetic.

All other values are reserved.

The permitted values are:

- 0000 in an implementation without floating-point support.
- 0010 in an implementation with floating-point support, that does not include the ARMv8.2-FP16 extension.
- 0011 in an implementation with floating-point support, that includes the ARMv8.2-FP16 extension.

### **SIMDHP, bits [23:20]**

Advanced SIMD Half Precision. Indicates the level of half-precision floating-point support. Defined values are:

<b>SIMDHP</b>	<b>Meaning</b>
0000	Not supported.
0001	SIMD half-precision conversion instructions are supported for conversion between single-precision and half-precision.
0010	As for 0010, and also includes support for half-precision floating-point arithmetic.

All other values are reserved.

The permitted values are:

- 0000 in an implementation without SIMD floating-point support.
- 0001 in an implementation with SIMD floating-point support, that does not include the ARMv8.2-FP16 extension.
- 0010 in an implementation with SIMD floating-point support, that includes the ARMv8.2-FP16 extension.

### **SIMDSP, bits [19:16]**

Advanced SIMD Single Precision. Indicates whether the Advanced SIMD and floating-point implementation provides single-precision floating-point instructions. Defined values are:

<b>SIMDSP</b>	<b>Meaning</b>
0000	Not implemented.
0001	Implemented. This value is permitted only if the SIMDInt field is 0001.

All other values are reserved.

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

### **SIMDInt, bits [15:12]**

Advanced SIMD Integer. Indicates whether the Advanced SIMD and floating-point implementation provides integer instructions. Defined values are:

<b>SIMDInt</b>	<b>Meaning</b>
0000	Not implemented.
0001	Implemented.

All other values are reserved.

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

**SIMDLS, bits [11:8]**

Advanced SIMD Load/Store. Indicates whether the Advanced SIMD and floating-point implementation provides load/store instructions. Defined values are:

SIMDLS	Meaning
0000	Not implemented.
0001	Implemented.

All other values are reserved.

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

**FPDNaN, bits [7:4]**

Default NaN mode. Indicates whether the floating-point implementation provides support only for the Default NaN mode. Defined values are:

FPDNaN	Meaning
0000	Not implemented, or hardware supports only the Default NaN mode.
0001	Hardware supports propagation of NaN values.

All other values are reserved.

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

**FPFtZ, bits [3:0]**

Flush to Zero mode. Indicates whether the floating-point implementation provides support only for the Flush-to-Zero mode of operation. Defined values are:

FPFtZ	Meaning
0000	Not implemented, or hardware supports only the Flush-to-Zero mode of operation.
0001	Hardware supports full denormalized number arithmetic.

All other values are reserved.

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

**Accessing the MVFR1**

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

```
VMRS <Rt>, <spec_reg>
```

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

<spec_reg>	reg
MVFR1	0110

**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 `HCR_EL2.E2H==0` :

- If `CPACR.cp10==00`, read accesses to this register from PL1 are UNDEFINED.

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

- If `CPTR_EL2.TFP==1`, Non-secure read accesses to this register from EL1 are trapped to EL2.
- 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 `CPTR_EL2.FPEN==00`, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If `CPTR_EL2.FPEN==10`, Non-secure read accesses to this register from EL1 are trapped to EL2.
- 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 AArch32 and `SCR_EL3.NS==1` :

- If `HCPTR.TCP10==1`, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.
- If `HCPTR.TCP10==1`, Non-secure read accesses to this register from EL2 are UNDEFINED.
- If `HCR.TID3==1`, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

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

- If `NSACR.cp10==0`, Non-secure read accesses to this register from EL1 and EL2 are UNDEFINED.

When EL3 is implemented and is using AArch64 :

- If `CPTR_EL3.TFP==1`, read accesses to this register from EL1 and EL2 are trapped to EL3.

02/05/2017 15:43

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

# MVFR2, Media and VFP Feature Register 2

The MVFR2 characteristics are:

## Purpose

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with [MVFR0](#) and [MVFR1](#).

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 Floating-point registers functional group.
- 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 MVFR2 is architecturally mapped to AArch64 System register [MVFR2\\_EL1](#).

Implemented only if the implementation includes Advanced SIMD and floating-point instructions.

## Attributes

MVFR2 is a 32-bit register.

## Field descriptions

The MVFR2 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	FPMisc				SIMDMisc			

### Bits [31:8]

Reserved, RES0.

### FPMisc, bits [7:4]

Indicates whether the floating-point implementation provides support for miscellaneous VFP features.

FPMisc	Meaning
0000	Not implemented, or no support for miscellaneous features.
0001	Support for Floating-point selection.
0010	As 0001, and Floating-point Conversion to Integer with Directed Rounding modes.
0011	As 0010, and Floating-point Round to Integer Floating-point.
0100	As 0011, and Floating-point MaxNum and MinNum.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0100.

**SIMDMisc, bits [3:0]**

Indicates whether the Advanced SIMD implementation provides support for miscellaneous Advanced SIMD features.

SIMDMisc	Meaning
0000	Not implemented, or no support for miscellaneous features.
0001	Floating-point Conversion to Integer with Directed Rounding modes.
0010	As 0001, and Floating-point Round to Integer Floating-point.
0011	As 0010, and Floating-point MaxNum and MinNum.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0011.

**Accessing the MVFR2**

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

```
VMRS <Rt>, <spec_reg>
```

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

<spec_reg>	reg
MVFR2	0101

**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<sup>®</sup> 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 [CPACR](#).cp10==00, read accesses to this register from PL1 are UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [CPTR\\_EL2](#).TFP==1, Non-secure read accesses to this register from EL1 are trapped to EL2.
- 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 [CPTR\\_EL2](#).FPEN==00, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [CPTR\\_EL2](#).FPEN==10, Non-secure read accesses to this register from EL1 are trapped to EL2.
- 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 AArch32 and SCR\_EL3.NS==1 :

- If [HCPTR.TCP10](#)==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.
- If [HCPTR.TCP10](#)==1, Non-secure read accesses to this register from EL2 are UNDEFINED.
- If [HCR.TID3](#)==1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

When EL3 is implemented and is using AArch32 and [SCR\\_EL3.NS](#)==1 :

- If [NSACR.cp10](#)==0, Non-secure read accesses to this register from EL1 and EL2 are UNDEFINED.

When EL3 is implemented and is using AArch64 :

- If [CPTR\\_EL3.TFP](#)==1, read accesses to this register from EL1 and EL2 are trapped to EL3.

02/05/2017 15:43

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



# NMRR, Normal Memory Remap Register

The NMRR characteristics are:

## Purpose

Provides additional mapping controls for memory regions that are mapped as Normal memory by their entry in the [PRRR](#).

Used in conjunction with the [PRRR](#).

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch32 System register NMRR is architecturally mapped to AArch64 System register [MAIR\\_EL1\[63:32\]](#) when TTBCR.EAE==0.

[MAIR1](#) and NMRR are the same register, with a different view depending on the value of [TTBCR](#).EAE:

- When it is set to 0, the register is as described in NMRR.
- When it is set to 1, the register is as described in [MAIR1](#).

When EL3 is using AArch32, write access to NMRR(S) is disabled when the CP15SDISABLE signal is asserted HIGH.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

NMRR is a 32-bit register.

## Field descriptions

The NMRR bit assignments are:

### When TTBCR.EAE==0:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<a href="#">OR7</a>	<a href="#">OR6</a>	<a href="#">OR5</a>	<a href="#">OR4</a>	<a href="#">OR3</a>	<a href="#">OR2</a>	<a href="#">OR1</a>	<a href="#">OR0</a>	<a href="#">IR7</a>	<a href="#">IR6</a>	<a href="#">IR5</a>	<a href="#">IR4</a>	<a href="#">IR3</a>	<a href="#">IR2</a>	<a href="#">IR1</a>	<a href="#">IR0</a>																

### OR<n>, bits [2n+17:2n+16], for n = 0 to 7

Outer Cacheable property mapping for memory attributes n, if the region is mapped as Normal memory by the PRRR.TR<n> entry. n is the value of the TEX[0], C, and B bits concatenated. The possible values of this field are:

OR<n>	Meaning
00	Region is Non-cacheable.
01	Region is Write-Back, Write-Allocate.
10	Region is Write-Through, no Write-Allocate.
11	Region is Write-Back, no Write-Allocate.

The meaning of the field with n = 6 is IMPLEMENTATION DEFINED and might differ from the meaning given here. This is because the meaning of the attribute combination {TEX[0] = 1, C = 1, B = 0} is IMPLEMENTATION DEFINED.

### IR<n>, bits [2n+1:2n], for n = 0 to 7

Inner Cacheable property mapping for memory attributes n, if the region is mapped as Normal memory by the PRRR.TR<n> entry. n is the value of the TEX[0], C, and B bits concatenated. The possible values of this field are:

IR<n>	Meaning
00	Region is Non-cacheable.
01	Region is Write-Back, Write-Allocate.
10	Region is Write-Through, no Write-Allocate.
11	Region is Write-Back, no Write-Allocate.

The meaning of the field with n = 6 is IMPLEMENTATION DEFINED and might differ from the meaning given here. This is because the meaning of the attribute combination {TEX[0] = 1, C = 1, B = 0} is IMPLEMENTATION DEFINED.

## Accessing the NMRR

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>, c10, c2, 1	000	001	1010	1111	0010

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	NMRR_s
EL3 not implemented	x	x	0	-	RW	n/a	n/a	NMRR
EL3 not implemented	x	0	1	-	RW	RW	n/a	NMRR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	NMRR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	NMRR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	NMRR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	NMRR
EL3 using AArch32	x	0	1	-	RW	RW	RW	NMRR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	NMRR_ns

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

When EL3 is using AArch32, write access to NMRR\_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.T10](#)==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](#).T10==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](#).T10==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# NSACR, Non-Secure Access Control Register

The NSACR characteristics are:

## Purpose

When EL3 is implemented and can use AArch32, defines the Non-secure access permissions to Trace, Advanced SIMD and floating-point functionality. Also includes IMPLEMENTATION DEFINED bits that can define Non-secure access permissions for IMPLEMENTATION DEFINED functionality.

This register is part of the Security registers functional group.

## Configuration

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

### Note

In AArch64 state, the NSACR controls are replaced by controls in [CPTR\\_EL3](#).

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

NSACR is a 32-bit register.

## Field descriptions

The NSACR 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	NSTRCDIS	0	IMPLEMENTATION DEFINED			NSASEDIS	0	0	0	cp11	cp10	0	0	0	0	0	0	0	0	0	0

If EL3 is implemented and is using AArch64 then:

- Any read of the NSACR from Non-secure EL2 or Non-secure EL1 returns a value of 0x00000C00.
- Any read or write to NSACR from Secure EL1 is trapped as an exception to EL3.

If EL3 is not implemented, then any read of the NSACR from EL2 or EL1 returns a value of 0x00000C00.

### Bits [31:21]

Reserved, RES0.

### NSTRCDIS, bit [20]

Disables Non-secure System register accesses to all implemented trace registers.

NSTRCDIS	Meaning
0	This control has no effect on: <ul style="list-style-type: none"> <li>System register access to implemented trace registers.</li> <li>The behavior of <a href="#">CPACR</a>.TRCDIS and <a href="#">HCPTR</a>.TTA.</li> </ul>
1	Non-secure System register accesses to all implemented trace registers are disabled, meaning: <ul style="list-style-type: none"> <li><a href="#">CPACR</a>.TRCDIS behaves as RAO/WI in Non-secure state, regardless of its actual value.</li> <li><a href="#">HCPTR</a>.TTA behaves as RAO/WI, regardless of its actual value.</li> </ul>

The implementation of this field must correspond to the implementation of the [CPACR](#).TRCDIS field:

- If [CPACR](#).TRCDIS is RAZ/WI, this field is RAZ/WI.
- If [CPACR](#).TRCDIS is RW, this field is RW.

#### Note

- The ETMv4 architecture does not permit EL0 to access the trace registers. If the implementation includes an ETMv4 implementation, EL0 accesses to the trace registers are UNDEFINED.
- The architecture does not provide Non-secure access controls on trace register accesses through the optional memory-mapped external debug interface.

System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### Bit [19]

Reserved, RES0.

#### IMPLEMENTATION DEFINED, bits [18:16]

IMPLEMENTATION DEFINED.

#### NSASEDIS, bit [15]

Disables Non-secure access to the Advanced SIMD functionality.

NSASEDIS	Meaning
0	This control has no effect on: <ul style="list-style-type: none"> <li>Non-secure access to Advanced SIMD functionality.</li> <li>The behavior of <a href="#">CPACR</a>.ASEDIS and <a href="#">HCPTR</a>.TASE.</li> </ul>
1	Non-secure access to the Advanced SIMD functionality is disabled, meaning: <ul style="list-style-type: none"> <li><a href="#">CPACR</a>.ASEDIS behaves as RAO/WI in Non-secure state, regardless of its actual value.</li> <li><a href="#">HCPTR</a>.TASE behaves as RAO/WI, regardless of its actual value.</li> </ul>

The implementation of this field must correspond to the implementation of the [CPACR](#).ASEDIS field:

- If [CPACR](#).ASEDIS is RES0, this field is RES0. If the implementation does not include Advanced SIMD and floating-point functionality, this field is RES0.
- If [CPACR](#).ASEDIS is RAZ/WI, this field is RAZ/WI.
- If [CPACR](#).ASEDIS is RW, this field is RW.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### Bits [14:12]

Reserved, RES0.

#### cp11, bit [11]

The value of this field is ignored. If this field is programmed with a different value to the cp10 field then this field is UNKNOWN on a direct read of the NSACR.

If the implementation does not include Advanced SIMD and floating-point functionality, this field is RES0.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to an architecturally UNKNOWN value.

### cp10, bit [10]

Enable Non-secure access to the Advanced SIMD and floating-point features. Possible values of the fields are:

cp10	Meaning
0	Advanced SIMD and floating-point features can be accessed only from Secure state. Any attempt to access this functionality from Non-secure state is UNDEFINED. When the PE is in Non-secure state: <ul style="list-style-type: none"> <li>The <a href="#">CPACR</a>.{cp11, cp10} fields ignore writes and read as 0b00, access denied.</li> <li>The <a href="#">HCPTR</a>.{TCP11, TCP10} fields behave as RAO/WI, regardless of their actual values.</li> </ul>
1	Advanced SIMD and floating-point features can be accessed from both Security states.

If Non-secure access to the Advanced SIMD and floating-point functionality is enabled, the [CPACR](#) must be checked to determine the level of access that is permitted.

The Advanced SIMD and floating-point features controlled by these fields are:

- Execution of any floating-point or Advanced SIMD instruction.
- Any access to the Advanced SIMD and floating-point registers D0-D31 and their views as S0-S31 and Q0-Q15.
- Any access to the [FPSCR](#), [FPSID](#), [MVFR0](#), [MVFR1](#), [MVFR2](#), or [FPEXC](#) System registers.

If the implementation does not include Advanced SIMD and floating-point functionality, this field is RES0.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to an architecturally UNKNOWN value.

### Bits [9:0]

Reserved, RES0.

## Accessing the NSACR

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, 2	000	010	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	-	RO	RO	RW
x	1	1	-	n/a	RO	RW

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

If EL3 is implemented and is using AArch64, any read from or write to NSACR from Secure EL1 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<sup>®</sup> 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T1](#)=1, Non-secure read 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 [HSTR\\_EL2.T1](#)=1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T1](#)=1, Non-secure read accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# PAR, Physical Address Register

The PAR characteristics are:

## Purpose

Returns the output address (OA) from an Address translation instruction that executed successfully, or fault information if the instruction did not execute successfully.

This register is part of the Address translation instructions functional group.

## Configuration

AArch32 System register PAR is architecturally mapped to AArch64 System register [PAR\\_EL1](#).

The PAR returns a 32-bit value:

- When the PE is not in Hyp mode and is using the Short-descriptor translation table format.
- When the PE is in Hyp mode and executes an [ATS12NSOPR](#), [ATS12NSOPW](#), [ATS12NSOUR](#), or [ATS12NSOUW](#) instruction when the value of [HCR.VM](#) is 0 and the value of [TTBCR.EAE](#) is 0.

In these cases, PAR[63:32] is RES0.

Otherwise, the PAR returns a 64-bit value. This means it returns a 64-bit value in the following cases:

- When using the Long-descriptor translation table format.
- If the stage 1 address translation is disabled and [TTBCR.EAE](#) is set to 1.
- In an implementation that includes EL2, for the result of an ATS1Cxx instruction performed from Hyp mode.

For PL1&0 stage 1 translations, [TTBCR.EAE](#) selects the translation table format.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

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

The Configurations section specifies the cases where each PAR format is used.

## Field descriptions

The PAR bit assignments are:

### For all register layouts:

#### F, bit [0]

Indicates whether the instruction performed a successful address translation.

F	Meaning
0	Address translation completed successfully.
1	Address translation aborted.

### When accessing PAR as a 32-bit register, PAR.F==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
PA											LPA	E	N	O	S	N	S	I	M	P	D	E	F	S	H	Inner[2:0]		Outer[1:0]		SS	F



This section describes the register value returned by the successful execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

On a successful conversion, the PAR can return a value that indicates the resulting attributes, rather than the values that appear in the translation table descriptors. More precisely:

- Memory attribute fields are permitted to report the resulting attributes, as determined by any permitted implementation choices and any applicable configuration bits, instead of reporting the values that appear in the translation table descriptors. This applies to the NOS, SH, Inner, and Outer fields.
- See the NS bit description for constraints on the value it returns.

### PA, bits [31:12]

Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[31:12].

### LPAE, bit [11]

When updating the PAR with the result of the translation operation, this bit is set as follows:

LPAE	Meaning
0	Short-descriptor translation table format used. This means the PAR returned a 32-bit value.

### NOS, bit [10]

Not Outer Shareable. When the returned value of PAR.SH is 1, indicates the Shareability attribute for the physical memory region:

NOS	Meaning
0	Memory region is Outer Shareable.
1	Memory region is Inner Shareable.

When the returned value of PAR.SH is 0 the value returned to this field is UNKNOWN.

The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.

### NS, bit [9]

Non-secure. The NS attribute for a translation table entry from a Secure translation regime.

For a result from a Secure translation regime, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation.

For a result from a Non-secure translation regime, this bit is UNKNOWN.

### IMP DEF, bit [8]

IMPLEMENTATION DEFINED.

### SH, bit [7]

Shareability. Indicates whether the physical memory region is Non-shareable:

SH	Meaning
0	Memory is Non-shareable.
1	Memory is shareable, and PAR.NOS indicates whether the region is Outer Shareable or Inner Shareable.

The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.

**Inner[2:0], bits [6:4]**

Inner cacheability attribute for the region. Permitted values are:

Inner	Meaning
000	Non-cacheable.
001	Device-nGnRnE.
011	Device-nGnRE.
101	Write-Back, Write-Allocate.
110	Write-Through.
111	Write-Back, no Write-Allocate.

The values 010 and 100 are reserved.

The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.

**Outer[1:0], bits [3:2]**

Outer cacheability attribute for the region. Permitted values are:

Outer	Meaning
00	Non-cacheable.
01	Write-Back, Write-Allocate.
10	Write-Through, no Write-Allocate.
11	Write-Back, no Write-Allocate.

The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.

**SS, bit [1]**

Supersection. Used to indicate if the result is a Supersection:

SS	Meaning
0	Result is not a Supersection. PAR[31:12] contains OA[31:12].
1	Result is a Supersection, and: <ul style="list-style-type: none"> <li>PAR[31:24] contains OA[31:24].</li> <li>PAR[23:16] contains OA[39:32].</li> <li>PAR[15:12] contains 0b0000.</li> </ul>

If an implementation supports less than 40 bits of physical address, the bits in the PAR field that correspond to physical address bits that are not implemented are UNKNOWN.

**F, bit [0]**

Indicates whether the instruction performed a successful address translation.

F	Meaning
0	Address translation completed successfully.

**When accessing PAR as a 32-bit register, PAR.F==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
IMP DEF																0	0	0	0	LPAE	0	0	0	0	FS				F		

This section describes the register value returned by a fault on the execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

**IMP DEF, bits [31:16]**

IMPLEMENTATION DEFINED.

**Bits [15:12]**

Reserved, RES0.

**LPAE, bit [11]**

When updating the PAR with the result of the translation operation, this bit is set as follows:

LPAE	Meaning
0	Short-descriptor translation table format used. This means the PAR returned a 32-bit value.

**Bits [10:7]**

Reserved, RES0.

**FS, bits [6:1]**Fault status bits. Bits [12,10,3:0] from the [DFSR](#), indicating the source of the abort.**F, bit [0]**

Indicates whether the instruction performed a successful address translation.

F	Meaning
1	Address translation aborted.

**When accessing PAR as a 64-bit register, PAR.F==0:**

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

This section describes the register value returned by the successful execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

On a successful conversion, the PAR can return a value that indicates the resulting attributes, rather than the values that appear in the translation table descriptors. More precisely:

- Memory attribute fields are permitted to report the resulting attributes, as determined by any permitted implementation choices and any applicable configuration bits, instead of reporting the values that appear in the translation table descriptors. This applies to the ATTR and SH fields.
- See the NS bit description for constraints on the value it returns.

**ATTR, bits [63:56]**Memory attributes for the returned output address. This field uses the same encoding as the Attr<n> fields in [MAIR0](#) and [MAIR1](#).

The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.

**Bits [55:40]**

Reserved, RES0.

**PA, bits [39:12]**

Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[39:12].

**LPAE, bit [11]**

When updating the PAR with the result of the translation operation, this bit is set as follows:

LPAE	Meaning
1	Long-descriptor translation table format used. This means the PAR returned a 64-bit value.

**IMP DEF, bit [10]**

IMPLEMENTATION DEFINED.

**NS, bit [9]**

Non-secure. The NS attribute for a translation table entry from a Secure translation regime.

For a result from a Secure translation regime, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation.

For a result from a Non-secure translation regime, this bit is UNKNOWN.

**SH, bits [8:7]**

Shareability attribute, for the returned output address. Permitted values are:

SH	Meaning
00	Non-shareable.
10	Outer Shareable.
11	Inner Shareable.

The value 01 is reserved.

**Note**

This field returns the value 10 for:

- Any type of Device memory.
- Normal memory with both Inner Non-cacheable and Outer Non-cacheable attributes.

The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.

**Bits [6:1]**

Reserved, RES0.

**F, bit [0]**

Indicates whether the instruction performed a successful address translation.

F	Meaning
0	Address translation completed successfully.

**When accessing PAR as a 64-bit register, PAR.F==1:**

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMP DEF								IMP DEF				IMP DEF				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	LPAE	0	FSTAGE	S2WLK	0	FST				F		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

This section describes the register value returned by a fault on the execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

**IMP DEF, bits [63:56]**

IMPLEMENTATION DEFINED.

**IMP DEF, bits [55:52]**

IMPLEMENTATION DEFINED.

**IMP DEF, bits [51:48]**

IMPLEMENTATION DEFINED.

**Bits [47:12]**

Reserved, RES0.

**LPAE, bit [11]**

When updating the PAR with the result of the translation operation, this bit is set as follows:

LPAE	Meaning
1	Long-descriptor translation table format used. This means the PAR returned a 64-bit value.

**Bit [10]**

Reserved, RES0.

**FSTAGE, bit [9]**

Indicates the translation stage at which the translation aborted:

FSTAGE	Meaning
0	Translation aborted because of a fault in the stage 1 translation.
1	Translation aborted because of a fault in the stage 2 translation.

**S2WLK, bit [8]**

If this bit is set to 1, it indicates the translation aborted because of a stage 2 fault during a stage 1 translation table walk.

**Bit [7]**

Reserved, RES0.

**FST, bits [6:1]**Fault status field. Values are as in the [DFSR.STATUS](#) and [IFSR.STATUS](#) fields when using the Long-descriptor translation table format.**F, bit [0]**

Indicates whether the instruction performed a successful address translation.

F	Meaning
1	Address translation aborted.

**Accessing the PAR**

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

MRC &lt;syntax&gt;

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

MCR &lt;syntax&gt;

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

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c7, c4, 0	000	000	0111	1111	0100

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

MRRC &lt;syntax&gt;

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

MCRR &lt;syntax&gt;

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

<syntax>	opc1	coproc	CRm
p15, 0, <Rt1>, <Rt2>, c7	0000	1111	0111

## 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	PAR
EL3 not implemented	x	0	1	-	RW	RW	n/a	PAR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	PAR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	PAR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	PAR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	PAR
EL3 using AArch32	x	0	1	-	RW	RW	RW	PAR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	PAR_ns
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	PAR_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<sup>®</sup> 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.T7==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.T7==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.T7==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.



# PMCCFILTR, Performance Monitors Cycle Count Filter Register

The PMCCFILTR characteristics are:

## Purpose

Determines the modes in which the Cycle Counter, [PMCCNTR](#), increments.

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 PMCCFILTR is architecturally mapped to AArch64 System register [PMCCFILTR\\_EL0](#).

AArch32 System register PMCCFILTR is architecturally mapped to External register [PMCCFILTR\\_EL0](#).

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

PMCCFILTR is a 32-bit register.

## Field descriptions

The PMCCFILTR 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
P	U	NSK	NSU	NSH	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

### P, bit [31]

Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are:

P	Meaning
0	Count cycles in EL1.
1	Do not count cycles in EL1.

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

### U, bit [30]

User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are:

U	Meaning
0	Count cycles in EL0.
1	Do not count cycles in EL0.

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

### NSK, bit [29]

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0.



If the value of this bit is equal to the value of P, cycles in Non-secure EL1 are counted.

Otherwise, cycles in Non-secure EL1 are not counted.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### NSU, bit [28]

Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of U, cycles in Non-secure EL0 are counted.

Otherwise, cycles in Non-secure EL0 are not counted.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### NSH, bit [27]

Non-secure EL2 (Hyp mode) filtering bit. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0.

NSH	Meaning
0	Do not count cycles in EL2.
1	Count cycles in EL2.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### Bits [26:0]

Reserved, RES0.

## Accessing the PMCCFILTR

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, c15, 7	000	111	1110	1111	1111

PMCCFILTR can also be accessed by using [PMXEVTYPER](#) with [PMSELR](#).SEL set to 0b11111.

## 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 :

- 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 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.

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.

02/05/2017 15:43

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

# 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\\_ELO](#) when accessing as a 64-bit register.

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

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<sup>®</sup> 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.

02/05/2017 15:43

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

# PMCEID0, Performance Monitors Common Event Identification register 0

The PMCEID0 characteristics are:

## Purpose

Defines which common architectural and common microarchitectural feature events in the range 0x000 to 0x01F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

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
																<a href="#">ID[31:0]</a>															

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

PMCEID0[n] maps to event n. For a list of event numbers and descriptions, see 'Event numbers and mnemonics' in the ARM ARM, section D5.10.

For each bit:

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

Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture.

Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.

## 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.

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 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.

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.

02/05/2017 15:43

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

# PMCEID1, Performance Monitors Common Event Identification register 1

The PMCEID1 characteristics are:

## Purpose

Defines which common architectural and common microarchitectural feature events in the range 0x020 to 0x03F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

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
<a href="#">ID[63:32]</a>																															

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

PMCEID1[n] maps to event (n + 32). For a list of event numbers and descriptions, see 'Event numbers and mnemonics' in the ARM ARM, section D5.10.

For each bit:

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

Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture.

Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.

## 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.

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 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.

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.

02/05/2017 15:43

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

# PMCEID2, Performance Monitors Common Event Identification register 2

The PMCEID2 characteristics are:

## Purpose

Defines which common architectural and common microarchitectural feature events in the range 0x4000 to 0x401F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

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
<a href="#">ID[16415:16384]</a>																															

### ID[16415:16384], bits [31:0]

PMCEID2[31:0] maps to common events 0x4000 to 0x401F. For a list of event numbers and descriptions, see 'Event numbers and mnemonics' in the ARM ARM, section D5.10.

For each bit:

ID[16415:16384]	Meaning
0	The common event is not implemented.
1	The common event is implemented.

Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture.

Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.

## 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.

02/05/2017 15:43

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

# PMCEID3, Performance Monitors Common Event Identification register 3

The PMCEID3 characteristics are:

## Purpose

Defines which common architectural and common microarchitectural feature events in the range 0x4020 to 0x403F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

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
ID[16447:16416]																															

### ID[16447:16416], bits [31:0]

PMCEID3[31:0] maps to common events 0x4020 to 0x403F. For a list of event numbers and descriptions, see 'Event numbers and mnemonics' in the ARM ARM, section D5.10.

For each bit:

ID[16447:16416]	Meaning
0	The common event is not implemented.
1	The common event is implemented.

Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture.

Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.

## 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.

02/05/2017 15:43

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

# 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 <a href="#">PMEVCNTR&lt;n&gt;</a> is disabled. When written, has no effect.
1	When read, means that <a href="#">PMEVCNTR&lt;n&gt;</a> is enabled. When written, disables <a href="#">PMEVCNTR&lt;n&gt;</a> .

## 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 [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.





# 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 <a href="#">PMEVCNTR&lt;n&gt;</a> is disabled. When written, has no effect.
1	When read, means that <a href="#">PMEVCNTR&lt;n&gt;</a> event counter is enabled. When written, enables <a href="#">PMEVCNTR&lt;n&gt;</a> .

## 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<sup>®</sup> 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.



# 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 [PMOVSr\[31\]](#).

LC	Meaning
0	Cycle counter overflow on increment that changes <a href="#">PMCCNTR[31]</a> from 1 to 0.
1	Cycle counter overflow on increment that changes <a href="#">PMCCNTR[63]</a> 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	<a href="#">PMCCNTR</a> , if enabled, counts when event counting is prohibited.
1	<a href="#">PMCCNTR</a> 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, <a href="#">PMCCNTR</a> counts every clock cycle.
1	When enabled, <a href="#">PMCCNTR</a> 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 <a href="#">PMCCNTR</a> 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 <a href="#">PMCCNTR</a> , 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 <a href="#">PMCCNTR</a> , are disabled.
1	All counters that are accessible at Non-secure EL1 are enabled by <a href="#">PMCNTENSET</a> .

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.

# PMEVCNTR<n>, Performance Monitors Event Count Registers, n = 0 - 30

The PMEVCNTR<n> characteristics are:

## Purpose

Holds event counter n, which counts events, where n is 0 to 30.

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 PMEVCNTR<n> is architecturally mapped to AArch64 System register [PMEVCNTR<n>\\_EL0](#).

AArch32 System register PMEVCNTR<n> is architecturally mapped to External register [PMEVCNTR<n>\\_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

PMEVCNTR<n> is a 32-bit register.

## Field descriptions

The PMEVCNTR<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
Event counter n																															

### Bits [31:0]

Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

## Accessing the PMEVCNTR<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
p15, 0, <Rt>, c14, <CRm>, <opc2>	000	n<2:0>	1110	1111	10:n<4:3>

- <opc2> is in the range 0 - 7.
- <CRm> is in the range c8 - c11.

PMEVCNTR<n> can also be accessed by using [PMXEVCNTR](#) with [PMSELR](#).SEL set to the value of <n>.



## 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 <n> is greater than or equal to the number of accessible counters, reads and writes of PMEVCNTR<n> are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- 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<sup>®</sup> 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 :

- 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 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.

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.



# PMEVTYPER<n>, Performance Monitors Event Type Registers, n = 0 - 30

The PMEVTYPER<n> characteristics are:

## Purpose

Configures event counter n, where n is 0 to 30.

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 PMEVTYPER<n> is architecturally mapped to AArch64 System register [PMEVTYPER<n>\\_EL0](#).

AArch32 System register PMEVTYPER<n> is architecturally mapped to External register [PMEVTYPER<n>\\_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

PMEVTYPER<n> is a 32-bit register.

## Field descriptions

The PMEVTYPER<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
P	U	NSK	NSU	NSH	0	MT	0	0	0	0	0	0	0	0	0	evtCount[15:10]						evtCount[9:0]									

### P, bit [31]

Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are:

P	Meaning
0	Count events in EL1.
1	Do not count events in EL1.

### U, bit [30]

User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are:

U	Meaning
0	Count events in EL0.
1	Do not count events in EL0.

### NSK, bit [29]

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted.

Otherwise, events in Non-secure EL1 are not counted.

**NSU, bit [28]**

Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted.

Otherwise, events in Non-secure EL0 are not counted.

**NSH, bit [27]**

Non-secure EL2 (Hyp mode) filtering bit. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0.

NSH	Meaning
0	Do not count events in EL2.
1	Count events in EL2.

**Bit [26]**

Reserved, RES0.

**MT, bit [25]**

Multithreading. When the implementation is multi-threaded, the valid values for this bit are:

MT	Meaning
0	Count events only on controlling PE.
1	Count events from any PE with the same affinity at level 1 and above as this PE.

When the implementation is not multi-threaded, this bit is RES0.

**Note**

- An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR\_EL1.MT, when read at the highest implemented Exception level, is 1.
- Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.

**Bits [24:16]**

Reserved, RES0.

**evtCount[15:10], bits [15:10]**

In ARMv8.2 and ARMv8.1:

Extension to evtCount[9:0]. See evtCount[9:0] for more details.

In ARMv8.0:

Reserved, RES0.

**evtCount[9:0], bits [9:0]**

Event to count. The event number of the event that is counted by event counter [PMEVCNTR<n>](#).

Software must program this field with an event that is supported by the PE being programmed.

There are three ranges of event numbers:

- Event numbers in the range 0x000 to 0x03F are common architectural and microarchitectural events.

- Event numbers in the range 0x040 to 0x0BF are ARM recommended common architectural and microarchitectural events.
- Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events.

If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type:

- For the range 0x000 to 0x03F, no events are counted, and the value returned by a direct or external read of the evtCount field is the value written to the field.
- For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the evtCount field is UNKNOWN.

---

#### Note

UNPREDICTABLE means the event must not expose privileged information.

---

ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

## Accessing the PMEVTYPER<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
p15, 0, <Rt>, c14, <CRm>, <opc2>	000	n<2:0>	1110	1111	11:n<4:3>

- <opc2> is in the range 0 - 7.
- <CRm> is in the range c12 - c15.

## Accessibility

The register is accessible as follows:

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

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

This register is accessible at EL0 when [PMUSERENR](#).EN is set to 1.

PMEVTYPER<n> can also be accessed by using [PMXEVTYPER](#) with [PMSELR](#).SEL set to n.

If <n> is greater or equal to the number of accessible counters, reads and writes of PMEVTYPER<n> are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- 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<sup>®</sup> 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 :

- 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 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.

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.

02/05/2017 15:43

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

# 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 <a href="#">PMEVCNTR&lt;n&gt;</a> event counter interrupt request is disabled. When written, has no effect.
1	When read, means that the <a href="#">PMEVCNTR&lt;n&gt;</a> event counter interrupt request is enabled. When written, disables the <a href="#">PMEVCNTR&lt;n&gt;</a> 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	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 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.





# 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 <a href="#">PMEVCNTR&lt;n&gt;</a> event counter interrupt request is disabled. When written, has no effect.
1	When read, means that the <a href="#">PMEVCNTR&lt;n&gt;</a> event counter interrupt request is enabled. When written, enables the <a href="#">PMEVCNTR&lt;n&gt;</a> 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	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 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.



# 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 <a href="#">PMEVCNTR&lt;n&gt;</a> has not overflowed. When written, has no effect.
1	When read, means that <a href="#">PMEVCNTR&lt;n&gt;</a> has overflowed. When written, clears the <a href="#">PMEVCNTR&lt;n&gt;</a> 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.



# 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 <a href="#">PMEVCNTR&lt;n&gt;</a> has not overflowed. When written, has no effect.
1	When read, means that <a href="#">PMEVCNTR&lt;n&gt;</a> has overflowed. When written, sets the <a href="#">PMEVCNTR&lt;n&gt;</a> 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<sup>®</sup> 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.



# 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, the results of access to [PMXEVTYPER](#) or [PMXEVNTR](#) are CONSTRAINED UNPREDICTABLE, and can be one of the following:

- Access to [PMXEVTYPER](#) or [PMXEVCNTR](#) is UNDEFINED.
- Access to [PMXEVTYPER](#) or [PMXEVCNTR](#) behaves as a NOP.
- Access to [PMXEVTYPER](#) or [PMXEVCNTR](#) behaves as if the register is RAZ/WI.
- Access to [PMXEVTYPER](#) or [PMXEVCNTR](#) behaves as if the PMSELR.SEL field contains an UNKNOWN value.
- Access to [PMXEVTYPER](#) or [PMXEVCNTR](#) 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>	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<sup>®</sup> 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.

02/05/2017 15:43

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

# 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 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.

The effects of writing to this bit are:

P<n>	Meaning
0	No action. The write to this bit is ignored.
1	If <a href="#">PMEVCNTR&lt;n&gt;</a> is enabled and configured to count the software increment event, increments <a href="#">PMEVCNTR&lt;n&gt;</a> by 1. If <a href="#">PMEVCNTR&lt;n&gt;</a> 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 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, 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	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<sup>®</sup> 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.

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.

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.

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.





# 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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	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	

### Bits [31:4]

Reserved, RES0.

### ER, bit [3]

Event counter read trap control:

ER	Meaning
0	EL0 reads of the <a href="#">PMXEVCNTR</a> and <a href="#">PMEVCNTR&lt;n&gt;</a> , and EL0 read/write access to the <a href="#">PMSELR</a> , 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 <a href="#">PMCCNTR</a> 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 <a href="#">PMSWINC</a> 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<sup>®</sup> 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 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 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.

02/05/2017 15:43

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

# 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
<a href="#">PMEVCNTR&lt;n&gt;</a>																															

### 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 PMXEVCNTR are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if [PMSELR.SEL](#) has an UNKNOWN value less than the number of 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<sup>®</sup> 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.

02/05/2017 15:43

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

# PMXEVTYPYPER, Performance Monitors Selected Event Type Register

The PMXEVTYPYPER characteristics are:

## Purpose

When [PMSELR.SEL](#) selects an event counter, this accesses a [PMEVTYPYPER<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 PMXEVTYPYPER is architecturally mapped to AArch64 System register [PMXEVTYPYPER\\_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

PMXEVTYPYPER is a 32-bit register.

## Field descriptions

The PMXEVTYPYPER 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 [PMEVTYPYPER<n>](#) where n is the value in [PMSELR.SEL](#).

## Accessing the PMXEVTYPYPER

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 PMXEVTYPER are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if [PMSELR](#).SEL has an UNKNOWN value less than the number of 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<sup>®</sup> 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.

02/05/2017 15:43

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

# PRRR, Primary Region Remap Register

The PRRR characteristics are:

## Purpose

Controls the top level mapping of the TEX[0], C, and B memory region attributes.

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch32 System register PRRR is architecturally mapped to AArch64 System register [MAIR\\_EL1\[31:0\]](#) when TTBCR.EAE==0.

[MAIRO](#) and PRRR are the same register, with a different view depending on the value of [TTBCR.EAE](#):

- When it is set to 0, the register is as described in PRRR.
- When it is set to 1, the register is as described in [MAIRO](#).

When EL3 is using AArch32, write access to PRRR(S) is disabled when the CP15SDISABLE signal is asserted HIGH.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

PRRR is a 32-bit register.

## Field descriptions

The PRRR bit assignments are:

### When TTBCR.EAE==0:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOS7	NOS6	NOS5	NOS4	NOS3	NOS2	NOS1	NOS0	0	0	0	0	NS1	NS0	DS1	DS0	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0								

### NOS<n>, bit [n+24], for n = 0 to 7

Not Outer Shareable. NOS<n> is the Outer Shareable property for memory attributes n, if the region is mapped as Normal memory that is not Inner Non-cacheable, Outer Non-cacheable, and the appropriate PRRR. {NS0, NS1} field identifies the region as shareable. n is the value of the concatenation of the {TEX[0], C, B} bits from the translation table descriptor. The possible values of each NOS<n> field other than NOS6 are:

NOS<n>	Meaning
0	Memory region is Outer Shareable.
1	Memory region is Inner Shareable.

The value of this bit is ignored if the region is:

- Device memory
- Normal memory that is at least one of:
  - Inner Non-cacheable, Outer Non-cacheable.
  - Identified by the appropriate PRRR. {NS0, NS1} field as Non-shareable.

The meaning of the NOS6 field is IMPLEMENTATION DEFINED.

### Bits [23:20]

Reserved, RES0.

**NS1, bit [19]**

Mapping of S = 1 attribute for Normal memory regions. This field is used in determining the Shareability of a memory region that is mapped to Normal memory and both:

- Is not Inner Non-cacheable, Outer Non-cacheable.
- Has the S bit in the translation table descriptor set to 1.

The possible values of this bit are:

NS1	Meaning
0	Region is Non-shareable.
1	Region is shareable. The value of the appropriate PRRR.NOS<n> field determines whether the region is Inner Shareable or Outer Shareable.

**NS0, bit [18]**

Mapping of S = 0 attribute for Normal memory regions. This field is used in determining the Shareability of a memory region that is mapped to Normal memory and both:

- Is not Inner Non-cacheable, Outer Non-cacheable.
- Has the S bit in the translation table descriptor set to 0.

The possible values of this bit are:

NS0	Meaning
0	Region is Non-shareable.
1	Region is shareable. The value of the appropriate PRRR.NOS<n> field determines whether the region is Inner Shareable or Outer Shareable.

**DS1, bit [17]**

Mapping of S = 1 attribute for Device memory. In ARMv8, all types of Device memory are Outer Shareable, and therefore this bit is RES1.

**DS0, bit [16]**

Mapping of S = 0 attribute for Device memory. In ARMv8, all types of Device memory are Outer Shareable, and therefore this bit is RES1.

**TR<n>, bits [2n+1:2n], for n = 0 to 7**

TR<n> is the primary TEX mapping for memory attributes n, and defines the mapped memory type for a region with attributes n. n is the value of the concatenation of the {TEX[0], C, B} bits from the translation table descriptor. The possible values for each field other than TR6 are:

TR<n>	Meaning
00	Device-nGnRnE memory
01	Device-nGnRE memory
10	Normal memory

The value 11 is reserved. The effect of programming a field to 11 is CONstrained UNPREDICTABLE, see 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARMv8 ARM, section K1.1.11.

The meaning of the TR6 field is IMPLEMENTATION DEFINED.

**Accessing the PRRR**

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>, c10, c2, 0	000	000	1010	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	n/a	n/a	PRRR
EL3 not implemented	x	0	1	-	RW	RW	n/a	PRRR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	PRRR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	PRRR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	PRRR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	PRRR
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	PRRR_s
EL3 using AArch32	x	0	1	-	RW	RW	RW	PRRR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	PRRR_ns

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

When EL3 is using AArch32, write access to PRRR\_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.T10](#)==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.T10](#)==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.T10](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

# REVIDR, Revision ID Register

The REVIDR characteristics are:

## Purpose

Provides implementation-specific minor revision information.

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 REVIDR is architecturally mapped to AArch64 System register [REVIDR\\_EL1](#).

If REVIDR has the same value as [MIDR](#), then its contents have no significance.

## Attributes

REVIDR is a 32-bit register.

## Field descriptions

The REVIDR 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the REVIDR

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, 6	000	110	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 [HCR\\_EL2](#).TID1==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](#).TID1==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](#).TID1==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.

02/05/2017 15:43

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

# RMR, Reset Management Register

The RMR characteristics are:

## Purpose

If EL1 or EL3 is the highest implemented Exception level and this register is implemented:

- A write to the register at the highest implemented Exception level can request a Warm reset.
- If the highest implemented Exception level can use AArch32 and AArch64, this register specifies the Execution state that the PE boots into on a Warm reset.

This register is part of the Reset management registers functional group.

## Configuration

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

AArch32 System register RMR is architecturally mapped to AArch64 System register [RMR\\_EL1](#) when EL1 is highest implemented Exception level.

AArch32 System register RMR is architecturally mapped to AArch64 System register [RMR\\_EL3](#) when EL3 is implemented.

Only implemented if EL1 or EL3 is the highest implemented Exception level. In this case:

- If the highest implemented Exception level can use AArch32 and AArch64 then this register must be implemented.
- If the highest implemented Exception level cannot use AArch64 then it is IMPLEMENTATION DEFINED whether the register is implemented.

See the field descriptions for the reset values. These apply whenever the register is implemented.

## Attributes

RMR is a 32-bit register.

## Field descriptions

The RMR 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	RRAA64

### Bits [31:2]

Reserved, RES0.

### RR, bit [1]

Reset Request. Setting this bit to 1 requests a Warm reset.

This field resets to 0 on a Warm or Cold reset.

### AA64, bit [0]

When the highest implemented Exception level can use AArch64, determines which Execution state the PE boots into after a Warm reset:

AA64	Meaning
0	AArch32.
1	AArch64.

On coming out of the Warm reset, execution starts at the IMPLEMENTATION DEFINED reset vector address of the specified Execution state.

If the highest implemented Exception level cannot use AArch64 this bit is RAZ/WI.

When implemented as a RW field, this field resets to 0 on a Cold reset.

## Accessing the RMR

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, c0, 2	000	010	1100	1111	0000

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
EL1 is the highest implemented Exception level	x	x	x	-	RW	n/a	n/a
EL2 is the highest implemented Exception level	x	0	1	-	-	-	n/a
EL2 is the highest implemented Exception level	x	1	1	-	n/a	-	n/a
EL3 is the highest implemented Exception level	x	x	0	-	-	n/a	RW
EL3 is the highest implemented Exception level	x	0	1	-	-	-	RW
EL3 is the highest implemented Exception level	x	1	1	-	n/a	-	RW

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

The encoding for this register is UNDEFINED:

- If the RMR is not implemented.
- At all Exception levels other than the highest implemented Exception level.

When EL3 is implemented, ARM deprecates accessing this register from any PE mode other than Monitor mode.



# RVBAR, Reset Vector Base Address Register

The RVBAR characteristics are:

## Purpose

If EL3 is not implemented, contains the IMPLEMENTATION DEFINED address that execution starts from after reset when executing in AArch32 state.

This register is part of the Reset management registers functional group.

## Configuration

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

This register is only implemented if the highest Exception level implemented is capable of using AArch32, and is not EL3.

## Attributes

RVBAR is a 32-bit register.

## Field descriptions

The RVBAR 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
Reset Address[31:1]																															1

### Bits [31:1]

Reset Address[31:1]. Bits [31:1] of the IMPLEMENTATION DEFINED address that execution starts from after reset when executing in 32-bit state.

### Bit [0]

Reserved, RES1.

## Accessing the RVBAR

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>, c12, c0, 1	000	001	1100	1111	0000

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
EL1 is the highest implemented Exception level	x	x	x	-	RO	n/a	n/a
EL2 is the highest implemented Exception level	x	0	1	-	-	RO	n/a

EL2 is the highest implemented Exception level	x	1	1	-	n/a	RO	n/a
EL3 is the highest implemented Exception level	x	x	0	-	-	n/a	-
EL3 is the highest implemented Exception level	x	0	1	-	-	-	-
EL3 is the highest implemented Exception level	x	1	1	-	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==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 AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==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.

02/05/2017 15:43

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

# 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	TWE	TWI	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	Does not trap accesses to record registers from EL1 and EL2 to EL3.
1	Accesses to the ER* registers from EL1 and EL2 generate a Trap 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 <a href="#">SCTLR.nTWE</a> or <a href="#">HCR.TWE</a> . 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 <a href="#">SCTLR.nTWI</a> or <a href="#">HCR.TWI</a> . 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"> <li>• UNDEFINED at Non-secure EL1. The Undefined Instruction exception is taken from PL1 to PL1.</li> <li>• UNPREDICTABLE at EL2. Behavior is one of the following: <ul style="list-style-type: none"> <li>◦ The instruction is UNDEFINED.</li> <li>◦ The instruction executes as a NOP.</li> </ul> </li> </ul>
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"> <li>• The instruction is UNDEFINED.</li> <li>• The instruction executes as a NOP.</li> </ul>

**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 <a href="#">CPSR.A</a> , and are taken to EL3.
1	External aborts taken from Secure state are masked by <a href="#">CPSR.A</a> . External aborts taken from either Security state are masked by <a href="#">CPSR.A</a> . When <a href="#">CPSR.A</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 <a href="#">CPSR.F</a> , and is taken to EL3. An FIQ taken from Secure state is masked by <a href="#">CPSR.F</a> .
1	An FIQ taken from either Security state is masked by <a href="#">CPSR.F</a> . When <a href="#">CPSR.F</a> 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<sup>®</sup> 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.

02/05/2017 15:43

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

# 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	AFE	TRE	0	0	EE	0	SPAN	1	0	UWXN	WXN	nTWE	0	nTWI	0	0	V	1	1	0	0	SEDI	TD	UNK	CP15	BEN	LSMA	OE	nTL	SMD	CAM

### 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 cleared to 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 ARMv8.2 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	<a href="#">CPSR</a> .PAN is set to 1 in the following situations: <ul style="list-style-type: none"> <li>In Non-secure state, on taking an exception to EL1.</li> <li>In Secure state, when EL3 is using AArch64, on taking an exception to EL1.</li> <li>In Secure state, when EL3 is using AArch32, on taking an exception to EL3.</li> </ul>
1	The value of <a href="#">CPSR</a> .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 <a href="#">VBAR</a> .
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:

<b>I</b>	<b>Meaning</b>
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.

<b>SED</b>	<b>Meaning</b>
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"> <li>All encodings of the IT instruction with <code>hw1[3:0] != 1000</code>.</li> <li>All encodings of the subsequent instruction with the following values for <code>hw1</code>: <div> <div>11xxxxxxxxxxxx</div> <div>All 32-bit instructions, and the 16-bit instructions B, UDF, SVC, LDM, and STM.</div> </div> <div> <div>1011xxxxxxxxxx</div> <div>All instructions in 'Miscellaneous 16-bit instructions' in the ARMv8 ARM, section F3.2.5.</div> </div> <div> <div>10100xxxxxxxxx</div> <div>ADD Rd, PC, #imm</div> </div> <div> <div>01001xxxxxxxxx</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> </li> </ul> <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"> <li>A 16-bit instruction, that can only be followed by another 16-bit instruction.</li> <li>The first half of a 32-bit instruction.</li> </ul> <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 <a href="#">CP15DMB</a> , <a href="#">CP15DSB</a> , and <a href="#">CP15ISB</a> instructions is UNDEFINED.
1	PL0 and PL1 execution of the <a href="#">CP15DMB</a> , <a href="#">CP15DSB</a> , and <a href="#">CP15ISB</a> 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 ARMv8.2:**

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 ARMv8.2:**

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.



# SDCR, Secure Debug Control Register

The SDCR characteristics are:

## Purpose

When EL3 is implemented and can use AArch32, controls debug and performance monitors functionality in Secure state.

This register is part of:

- The Debug registers functional group.
- The Security registers functional group.

## Configuration

This register is only accessible in Secure state.

AArch32 System register SDCR can be mapped to AArch64 System register [MDCR\\_EL3](#), but this is not architecturally mandated.

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

SDCR is a 32-bit register.

## Field descriptions

The SDCR 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	EPMAD	EDAD	0	0	SPME	0	SPD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### Bits [31:22]

Reserved, RES0.

### EPMAD, bit [21]

External debug interface Performance Monitors registers disable. This disables access to these registers by an external debugger:

EPMAD	Meaning
0	Access to Performance Monitors registers from external debugger is permitted.
1	Access to Performance Monitors registers from external debugger is disabled, unless overridden by the IMPLEMENTATION DEFINED authentication interface.

If the Performance Monitors Extension is not implemented or does not support external debug interface accesses 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.

### EDAD, bit [20]

External debug interface breakpoint and watchpoint register access disable. This disables access to these registers by an external debugger:

EDAD	Meaning
0	Access to breakpoint and watchpoint registers from external debugger is permitted.
1	Access to breakpoint and watchpoint registers from external debugger is disabled, unless overridden by the IMPLEMENTATION DEFINED authentication interface.

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

#### Bits [19:18]

Reserved, RES0.

#### SPME, bit [17]

Secure Performance Monitors enable. This allows event counting in Secure state:

SPME	Meaning
0	Event counting prohibited in Secure state. In an ARMv8.0 or ARMv8.1 implementation, event counting is prohibited unless ExternalSecureNoninvasiveDebugEnabled() is TRUE, meaning this control is overridden by the IMPLEMENTATION DEFINED authentication interface.
1	Event counting allowed in Secure state.

If the Performance Monitors Extension is not implemented, this field is RES0.

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

#### Bit [16]

Reserved, RES0.

#### SPD, bits [15:14]

AArch32 Secure privileged debug. Enables or disables debug exceptions from Secure state, other than Breakpoint Instruction exceptions. Valid values for this field are:

SPD	Meaning
00	Legacy mode. Debug exceptions from Secure EL1 are enabled by the authentication interface.
10	Secure privileged debug disabled. Debug exceptions from Secure EL1 are disabled.
11	Secure privileged debug enabled. Debug exceptions from Secure EL1 are enabled.

Other values are reserved, and have the CONSTRAINED UNPREDICTABLE behavior that they must have the same behavior as 0b00. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

If debug exceptions from Secure EL1 are enabled, then debug exceptions from Secure EL0 are also enabled.

Otherwise, debug exceptions from Secure EL0 are enabled only if [SDER32\\_EL3.SUIDEN](#) == 1.

Ignored in Non-secure state. Debug exceptions from Breakpoint Instruction exceptions are always enabled.

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

#### Bits [13:0]

Reserved, RES0.

## Accessing the SDCR

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 &lt;syntax&gt;

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

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c1, c3, 1	000	001	0001	1111	0011

## 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 SDCR 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<sup>®</sup> 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.

# SDER, Secure Debug Enable Register

The SDER characteristics are:

## Purpose

Controls invasive and non-invasive debug in the Secure EL0 mode.

This register is part of:

- The Debug registers functional group.
- The Security registers functional group.

## Configuration

This register is only accessible in Secure state.

AArch32 System register SDER is architecturally mapped to AArch64 System register [SDER32\\_EL3](#).

If EL3 is not implemented and EL1 supports AArch32, SDER is implemented only if the implemented Security state is Secure state.

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

SDER is a 32-bit register.

## Field descriptions

The SDER 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	<a href="#">SUNIDEN</a>	<a href="#">SUIDEN</a>

### Bits [31:2]

Reserved, RES0.

### SUNIDEN, bit [1]

Secure User Non-Invasive Debug Enable:

SUNIDEN	Meaning
0	Performance Monitors event counting prohibited in Secure EL0 unless allowed by <a href="#">MDCR_EL3.SPME</a> , <a href="#">SDCR.SPME</a> . In an ARMv8.0 or ARMv8.1 implementation, event counting can also be allowed using the IMPLEMENTATION DEFINED authentication interface ExternalSecureNoninvasiveDebugEnabled().
1	Performance Monitors event counting allowed in Secure EL0.

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

### SUIDEN, bit [0]

Secure User Invasive Debug Enable:

SUIDEN	Meaning
0	Debug exceptions other than Breakpoint Instruction exceptions from Secure EL0 are disabled, unless enabled by <a href="#">MDCR_EL3.SPD32</a> or <a href="#">SDCR.SPD</a> .
1	Debug exceptions from Secure EL0 are enabled.

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

## Accessing the SDER

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, 1	000	001	0001	1111	0001

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
EL3 not implemented	x	x	0	-	RW	n/a	n/a
EL3 not implemented	x	0	1	-	-	-	n/a
EL3 not implemented	x	1	1	-	n/a	-	n/a
EL3 using AArch64	x	x	0	-	RW	n/a	n/a
EL3 using AArch64	x	0	1	-	-	-	n/a
EL3 using AArch64	x	1	1	-	n/a	-	n/a
EL3 using AArch32	x	x	0	-	n/a	n/a	RW
EL3 using AArch32	x	0	1	-	-	-	RW
EL3 using AArch32	x	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 G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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.



# SPSR, Saved Program Status Register

The SPSR characteristics are:

## Purpose

Holds the saved process state for the current mode.

This register is part of the Special-purpose registers functional group.

## Usage constraints

The SPSR can be read using the MRS instruction and written using the MSR (immediate) or MSR (register) instructions. For more details on the instruction syntax, see 'PSTATE and banked register access instructions' in the ARMv8 ARM, section F1.5.

## Traps and Enables

There are no traps or enables affecting this register.

## Configuration

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

## Attributes

SPSR is a 32-bit register.

## Field descriptions

The SPSR 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
N	Z	C	V	Q	IT[1:0]	J	0	PAN	0	IL		GE				IT[7:2]					E	A	I	F	T	M[4]		M[3:0]			

### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to the current mode, and copied to [CPSR.N](#) on executing an exception return operation in the current mode.

### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to the current mode, and copied to [CPSR.Z](#) on executing an exception return operation in the current mode.

### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to the current mode, and copied to [CPSR.C](#) on executing an exception return operation in the current mode.

### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to the current mode, and copied to [CPSR.V](#) on executing an exception return operation in the current mode.

**Q, bit [27]**

Set to the value of [CPSR.Q](#) on taking an exception to the current mode, and copied to [CPSR.Q](#) on executing an exception return operation in the current mode.

**IT[1:0], bits [26:25]**

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]**

In ARMv8.2 and ARMv8.1:

When ARMv8.1-PAN is implemented, set to the value of [CPSR.PAN](#) on taking an exception to the current mode, and copied to [CPSR.PAN](#) on executing an exception return operation in the current mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

In ARMv8.0:

Reserved, RES0.

**Bit [21]**

Reserved, RES0.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:



<b>E</b>	<b>Meaning</b>
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTLR.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### **A, bit [8]**

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

<b>A</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

#### **I, bit [7]**

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

<b>I</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

#### **F, bit [6]**

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

<b>F</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

#### **T, bit [5]**

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

<b>T</b>	<b>Meaning</b>
0	Taken from A32 state.
1	Taken from T32 state.

#### **M[4], bit [4]**

Execution state that the exception was taken from. Possible values of this bit are:

<b>M[4]</b>	<b>Meaning</b>
1	Exception taken from AArch32.

#### **M[3:0], bits [3:0]**

AArch32 mode that an exception was taken from. The possible values are:

<b>M[3:0]</b>	<b>Mode</b>
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor (only valid in Secure state, if EL3 is implemented and can use AArch32)
0b0111	Abort
0b1010	Hyp
0b1011	Undefined
0b1111	System

Other values are reserved.

02/05/2017 15:43

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

# SPSR\_abt, Saved Program Status Register (Abort mode)

The SPSR\_abt characteristics are:

## Purpose

Holds the saved process state when an exception is taken to Abort mode.

This register is part of the Special-purpose registers functional group.

## Configuration

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

AArch32 System register SPSR\_abt is architecturally mapped to AArch64 System register [SPSR\\_abt](#).

## Attributes

SPSR\_abt is a 32-bit register.

## Field descriptions

The SPSR\_abt 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
N	Z	C	V	Q	IT[1:0]	J	0	PAN	0	IL			GE					IT[7:2]				E	A	I	F	T	M[4]		M[3:0]		

### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to Abort mode, and copied to [CPSR.N](#) on executing an exception return operation in Abort mode.

### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to Abort mode, and copied to [CPSR.Z](#) on executing an exception return operation in Abort mode.

### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to Abort mode, and copied to [CPSR.C](#) on executing an exception return operation in Abort mode.

### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to Abort mode, and copied to [CPSR.V](#) on executing an exception return operation in Abort mode.

### Q, bit [27]

Set to the value of [CPSR.Q](#) on taking an exception to Abort mode, and copied to [CPSR.Q](#) on executing an exception return operation in Abort mode.

### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR](#).PAN on taking an exception to Abort mode, and copied to [CPSR](#).PAN on executing an exception return operation in Abort mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**Bit [21]**

Reserved, RES0.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTLR.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

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

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

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

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor (only valid in Secure state, if EL3 is implemented and can use AArch32)
0b0111	Abort
0b1011	Undefined
0b1111	System

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

## Accessing the SPSR\_abt

This register can be read using MRS (banked register) with the following syntax:

```
MRS <Rd>, <banked_reg>
```

This register can be written using MSR (banked register) with the following syntax:

```
MSR <banked_reg>, <Rd>
```

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

<banked_reg>	R	M	M1
SPSR_abt	1	1	0100

## 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.

Using MRS (banked register) and MSR (banked register) instructions, at PL1 this register is only accessible from PE modes other than Abort mode. In Abort mode, it is accessible as the current SPSR.

02/05/2017 15:43

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

# SPSR\_fiq, Saved Program Status Register (FIQ mode)

The SPSR\_fiq characteristics are:

## Purpose

Holds the saved process state when an exception is taken to FIQ mode.

This register is part of the Special-purpose registers functional group.

## Configuration

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

AArch32 System register SPSR\_fiq is architecturally mapped to AArch64 System register [SPSR\\_fiq](#).

## Attributes

SPSR\_fiq is a 32-bit register.

## Field descriptions

The SPSR\_fiq 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
N	Z	C	V	Q	IT[1:0]	J	0	PAN	0	IL			GE					IT[7:2]				E	A	I	F	T	M[4]		M[3:0]		

### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to FIQ mode, and copied to [CPSR.N](#) on executing an exception return operation in FIQ mode.

### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to FIQ mode, and copied to [CPSR.Z](#) on executing an exception return operation in FIQ mode.

### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to FIQ mode, and copied to [CPSR.C](#) on executing an exception return operation in FIQ mode.

### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to FIQ mode, and copied to [CPSR.V](#) on executing an exception return operation in FIQ mode.

### Q, bit [27]

Set to the value of [CPSR.Q](#) on taking an exception to FIQ mode, and copied to [CPSR.Q](#) on executing an exception return operation in FIQ mode.

### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

### J, bit [24]

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

Bit [23]

Reserved, RES0.

PAN, bit [22]

In ARMv8.2 and ARMv8.1:

When ARMv8.1-PAN is implemented, set to the value of CPSR.PAN on taking an exception to FIQ mode, and copied to CPSR.PAN on executing an exception return operation in FIQ mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

In ARMv8.0:

Reserved, RES0.

Bit [21]

Reserved, RES0.

IL, bit [20]

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

GE, bits [19:16]

Greater than or Equal flags, for parallel addition and subtraction.

IT[7:2], bits [15:10]

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

E, bit [9]

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTL.R.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.



**A, bit [8]**

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

**I, bit [7]**

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

I	Meaning
0	Exception not masked.
1	Exception masked.

**F, bit [6]**

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

F	Meaning
0	Exception not masked.
1	Exception masked.

**T, bit [5]**

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

**M[4], bit [4]**

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

**M[3:0], bits [3:0]**

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor (only valid in Secure state, if EL3 is implemented and can use AArch32)
0b0111	Abort
0b1011	Undefined
0b1111	System

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

## Accessing the SPSR\_fiq

This register can be read using MRS (banked register) with the following syntax:

```
MRS <Rd>, <banked_reg>
```

This register can be written using MSR (banked register) with the following syntax:

```
MSR <banked_reg>, <Rd>
```

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

<banked_reg>	R	M	M1
SPSR_fiq	1	0	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.

Using MRS (banked register) and MSR (banked register) instructions, at PL1 this register is only accessible from PE modes other than FIQ mode. In FIQ mode, it is accessible as the current SPSR.

02/05/2017 15:43

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

# SPSR\_hyp, Saved Program Status Register (Hyp mode)

The SPSR\_hyp characteristics are:

## Purpose

Holds the saved process state when an exception is taken to Hyp mode.

This register is part of the Special-purpose registers functional group.

## Configuration

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

AArch32 System register SPSR\_hyp is architecturally mapped to AArch64 System register [SPSR\\_EL2](#).

## Attributes

SPSR\_hyp is a 32-bit register.

## Field descriptions

The SPSR\_hyp 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
N	Z	C	V	Q	IT[1:0]	J	0	PAN	0	IL		GE						IT[7:2]				E	A	I	F	T	M[4]		M[3:0]		

### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to Hyp mode, and copied to [CPSR.N](#) on executing an exception return operation in Hyp mode.

### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to Hyp mode, and copied to [CPSR.Z](#) on executing an exception return operation in Hyp mode.

### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to Hyp mode, and copied to [CPSR.C](#) on executing an exception return operation in Hyp mode.

### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to Hyp mode, and copied to [CPSR.V](#) on executing an exception return operation in Hyp mode.

### Q, bit [27]

Set to the value of [CPSR.Q](#) on taking an exception to Hyp mode, and copied to [CPSR.Q](#) on executing an exception return operation in Hyp mode.

### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR](#).PAN on taking an exception to Hyp mode, and copied to [CPSR](#).PAN on executing an exception return operation in Hyp mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**Bit [21]**

Reserved, RES0.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTL.R.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

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

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

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

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that an exception was taken from. The possible values are:

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

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

## Accessing the SPSR\_hyp

This register can be read using MRS (banked register) with the following syntax:

```
MRS <Rd>, <banked_reg>
```

This register can be written using MSR (banked register) with the following syntax:

```
MSR <banked_reg>, <Rd>
```

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

<banked_reg>	R	M	M1
SPSR_hyp	1	1	1110

## 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.

Using MRS (banked register) and MSR (banked register) instructions, this register is only accessible from Monitor mode. In Hyp mode, this register is accessible as the current SPSR.

02/05/2017 15:43

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

# SPSR\_irq, Saved Program Status Register (IRQ mode)

The SPSR\_irq characteristics are:

## Purpose

Holds the saved process state when an exception is taken to IRQ mode.

This register is part of the Special-purpose registers functional group.

## Configuration

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

AArch32 System register SPSR\_irq is architecturally mapped to AArch64 System register [SPSR\\_irq](#).

## Attributes

SPSR\_irq is a 32-bit register.

## Field descriptions

The SPSR\_irq 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
N	Z	C	V	Q	IT[1:0]	J	0	PAN	0	IL	GE					IT[7:2]					E		A	I	F	T	M[4]	M[3:0]			

### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to IRQ mode, and copied to [CPSR.N](#) on executing an exception return operation in IRQ mode.

### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to IRQ mode, and copied to [CPSR.Z](#) on executing an exception return operation in IRQ mode.

### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to IRQ mode, and copied to [CPSR.C](#) on executing an exception return operation in IRQ mode.

### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to IRQ mode, and copied to [CPSR.V](#) on executing an exception return operation in IRQ mode.

### Q, bit [27]

Set to the value of [CPSR.Q](#) on taking an exception to IRQ mode, and copied to [CPSR.Q](#) on executing an exception return operation in IRQ mode.

### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR.PAN](#) on taking an exception to IRQ mode, and copied to [CPSR.PAN](#) on executing an exception return operation in IRQ mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**Bit [21]**

Reserved, RES0.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTL.R.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.



If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

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

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

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

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor (only valid in Secure state, if EL3 is implemented and can use AArch32)
0b0111	Abort
0b1011	Undefined
0b1111	System

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

## Accessing the SPSR\_irq

This register can be read using MRS (banked register) with the following syntax:

```
MRS <Rd>, <banked_reg>
```

This register can be written using MSR (banked register) with the following syntax:

```
MSR <banked_reg>, <Rd>
```

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

<banked_reg>	R	M	M1
SPSR_irq	1	1	0000

## 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.

Using MRS (banked register) and MSR (banked register) instructions, at PL1 this register is only accessible from PE modes other than IRQ mode. In IRQ mode, it is accessible as the current SPSR.

02/05/2017 15:43

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

# SPSR\_mon, Saved Program Status Register (Monitor mode)

The SPSR\_mon characteristics are:

## Purpose

Holds the saved process state when an exception is taken to Monitor mode.

This register is part of the Special-purpose registers functional group.

## Configuration

This register is only accessible in Secure state.

AArch32 System register SPSR\_mon can be mapped to AArch64 System register [SPSR\\_EL3](#), but this is not architecturally mandated.

## Attributes

SPSR\_mon is a 32-bit register.

## Field descriptions

The SPSR\_mon 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
N	Z	C	V	Q	IT[1:0]	J	0	PAN	0	IL			GE					IT[7:2]				E	A	I	F	T	M[4]		M[3:0]		

### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to Monitor mode, and copied to [CPSR.N](#) on executing an exception return operation in Monitor mode.

### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to Monitor mode, and copied to [CPSR.Z](#) on executing an exception return operation in Monitor mode.

### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to Monitor mode, and copied to [CPSR.C](#) on executing an exception return operation in Monitor mode.

### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to Monitor mode, and copied to [CPSR.V](#) on executing an exception return operation in Monitor mode.

### Q, bit [27]

Set to the value of [CPSR.Q](#) on taking an exception to Monitor mode, and copied to [CPSR.Q](#) on executing an exception return operation in Monitor mode.

### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR.PAN](#) on taking an exception to Monitor mode, and copied to [CPSR.PAN](#) on executing an exception return operation in Monitor mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**Bit [21]**

Reserved, RES0.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTL.R.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

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

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

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

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor (only valid in Secure state, if EL3 is implemented and can use AArch32)
0b0111	Abort
0b1010	Hyp
0b1011	Undefined
0b1111	System

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

## Accessing the SPSR\_mon

This register can be read using MRS (banked register) with the following syntax:

```
MRS <Rd>, <banked_reg>
```

This register can be written using MSR (banked register) with the following syntax:

```
MSR <banked_reg>, <Rd>
```

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

<banked_reg>	R	M	M1
SPSR_mon	1	1	1100

## 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.

Using MRS (banked register) and MSR (banked register) instructions, this register is only accessible from EL3 modes other than Monitor mode. In Monitor mode, it is accessible as the current SPSR.

02/05/2017 15:43

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

# SPSR\_svc, Saved Program Status Register (Supervisor mode)

The SPSR\_svc characteristics are:

## Purpose

Holds the saved process state when an exception is taken to Supervisor mode.

This register is part of the Special-purpose registers functional group.

## Configuration

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

AArch32 System register SPSR\_svc is architecturally mapped to AArch64 System register [SPSR\\_EL1](#).

## Attributes

SPSR\_svc is a 32-bit register.

## Field descriptions

The SPSR\_svc 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
N	Z	C	V	Q	IT[1:0]	J	0	PAN	0	IL	GE					IT[7:2]					E		A	I	F	T	M[4]	M[3:0]			

### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to Supervisor mode, and copied to [CPSR.N](#) on executing an exception return operation in Supervisor mode.

### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to Supervisor mode, and copied to [CPSR.Z](#) on executing an exception return operation in Supervisor mode.

### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to Supervisor mode, and copied to [CPSR.C](#) on executing an exception return operation in Supervisor mode.

### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to Supervisor mode, and copied to [CPSR.V](#) on executing an exception return operation in Supervisor mode.

### Q, bit [27]

Set to the value of [CPSR.Q](#) on taking an exception to Supervisor mode, and copied to [CPSR.Q](#) on executing an exception return operation in Supervisor mode.

### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR](#).PAN on taking an exception to Supervisor mode, and copied to [CPSR](#).PAN on executing an exception return operation in Supervisor mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**Bit [21]**

Reserved, RES0.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTLR.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.



If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

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

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

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

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor (only valid in Secure state, if EL3 is implemented and can use AArch32)
0b0111	Abort
0b1011	Undefined
0b1111	System

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

## Accessing the SPSR\_svc

This register can be read using MRS (banked register) with the following syntax:

```
MRS <Rd>, <banked_reg>
```

This register can be written using MSR (banked register) with the following syntax:

```
MSR <banked_reg>, <Rd>
```

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

<banked_reg>	R	M	M1
SPSR_svc	1	1	0010

## 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.

Using MRS (banked register) and MSR (banked register) instructions, at PL1 this register is only accessible from PE modes other than Supervisor mode. In Supervisor mode, it is accessible as the current SPSR.

02/05/2017 15:43

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

# SPSR\_und, Saved Program Status Register (Undefined mode)

The SPSR\_und characteristics are:

## Purpose

Holds the saved process state when an exception is taken to Undefined mode.  
This register is part of the Special-purpose registers functional group.

## Configuration

There is one instance of this register that is used in both Secure and Non-secure states.  
AArch32 System register SPSR\_und is architecturally mapped to AArch64 System register [SPSR\\_und](#).

## Attributes

SPSR\_und is a 32-bit register.

## Field descriptions

The SPSR\_und 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
N	Z	C	V	Q	IT[1:0]	J	0	PAN	0	IL	GE		IT[7:2]				E	A	I	F	T	M[4]	M[3:0]								

### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to Undefined mode, and copied to [CPSR.N](#) on executing an exception return operation in Undefined mode.

### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to Undefined mode, and copied to [CPSR.Z](#) on executing an exception return operation in Undefined mode.

### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to Undefined mode, and copied to [CPSR.C](#) on executing an exception return operation in Undefined mode.

### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to Undefined mode, and copied to [CPSR.V](#) on executing an exception return operation in Undefined mode.

### Q, bit [27]

Set to the value of [CPSR.Q](#) on taking an exception to Undefined mode, and copied to [CPSR.Q](#) on executing an exception return operation in Undefined mode.

### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR.PAN](#) on taking an exception to Undefined mode, and copied to [CPSR.PAN](#) on executing an exception return operation in Undefined mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**Bit [21]**

Reserved, RES0.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTL.R.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

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

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

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

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor (only valid in Secure state, if EL3 is implemented and can use AArch32)
0b0111	Abort
0b1011	Undefined
0b1111	System

Other values are reserved. The effect of programming this field to a Reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

## Accessing the SPSR\_und

This register can be read using MRS (banked register) with the following syntax:

```
MRS <Rd>, <banked_reg>
```

This register can be written using MSR (banked register) with the following syntax:

```
MSR <banked_reg>, <Rd>
```

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

<banked_reg>	R	M	M1
SPSR_und	1	1	0110

## 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.

Using MRS (banked register) and MSR (banked register) instructions, at PL1 this register is only accessible from PE modes other than Undefined mode. In Undefined mode, it is accessible as the current SPSR.

02/05/2017 15:43

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

# TCMTR, TCM Type Register

The TCMTR characteristics are:

## Purpose

Provides information about the implementation of the TCM.

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.

## Attributes

TCMTR is a 32-bit register.

## Field descriptions

The TCMTR 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the TCMTR

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, 2	000	010	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 [HCR\\_EL2](#).TID1==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](#).TID1==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](#).TID1==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.

02/05/2017 15:43

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



# TLBTR, TLB Type Register

The TLBTR characteristics are:

## Purpose

Provides information about the TLB implementation. The register must define whether the implementation provides separate instruction and data TLBs, or a unified TLB. Normally, the IMPLEMENTATION DEFINED information in this register includes the number of lockable entries in the TLB.

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.

## Attributes

TLBTR is a 32-bit register.

## Field descriptions

The TLBTR 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
IMPLEMENTATION DEFINED																															nU

### IMPLEMENTATION DEFINED, bits [31:1]

IMPLEMENTATION DEFINED.

### nU, bit [0]

Not Unified TLB. Indicates whether the implementation has a unified TLB:

nU	Meaning
0	Unified TLB.
1	Separate Instruction and Data TLBs.

## Accessing the TLBTR

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, 3	000	011	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 [HCR\\_EL2](#).TID1==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](#).TID1==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](#).TID1==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.

02/05/2017 15:43

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

# TPIDRPRW, PL1 Software Thread ID Register

The TPIDRPRW characteristics are:

## Purpose

Provides a location where software executing at EL1 or higher can store thread identifying information that is not visible to software executing at EL0, for OS management purposes.

The PE makes no use of this register.

This register is part of the Thread and process ID registers functional group.

## Configuration

AArch32 System register TPIDRPRW is architecturally mapped to AArch64 System register [TPIDR\\_EL1\[31:0\]](#).

The PE never updates this register. This means the register is always UNKNOWN on reset.

## Attributes

TPIDRPRW is a 32-bit register.

## Field descriptions

The TPIDRPRW 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
Thread ID																															

### Bits [31:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

## Accessing the TPIDRPRW

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>, c13, c0, 4	000	100	1101	1111	0000

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	TPIDRPRW_s

EL3 not implemented	x	x	0	-	RW	n/a	n/a	TPIDRPRW
EL3 not implemented	x	0	1	-	RW	RW	n/a	TPIDRPRW
EL3 not implemented	x	1	1	-	n/a	RW	n/a	TPIDRPRW
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	TPIDRPRW
EL3 using AArch64	x	0	1	-	RW	RW	n/a	TPIDRPRW
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	TPIDRPRW
EL3 using AArch32	x	0	1	-	RW	RW	RW	TPIDRPRW_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	TPIDRPRW_ns

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.T13==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.T13==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.T13==1](#), Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# TPIDRURO, PL0 Read-Only Software Thread ID Register

The TPIDRURO characteristics are:

## Purpose

Provides a location where software executing at EL1 or higher can store thread identifying information that is visible to software executing at EL0, for OS management purposes.

The PE makes no use of this register.

This register is part of the Thread and process ID registers functional group.

## Configuration

AArch32 System register TPIDRURO is architecturally mapped to AArch64 System register [TPIDRRO\\_EL0\[31:0\]](#).

The PE never updates this register. This means the register is always UNKNOWN on reset.

## Attributes

TPIDRURO is a 32-bit register.

## Field descriptions

The TPIDRURO 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
Thread ID																															

### Bits [31:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

## Accessing the TPIDRURO

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>, c13, c0, 3	000	011	1101	1111	0000

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	
EL3 using AArch32	x	x	0	RO	n/a	n/a	RW	TPIDRURO_s

EL3 using AArch32	x	0	1	RO	RW	RW	RW	TPIDRURO_ns
EL3 using AArch32	x	1	1	RO	n/a	RW	RW	TPIDRURO_ns
EL3 not implemented	x	x	0	RO	RW	n/a	n/a	TPIDRURO
EL3 not implemented	x	0	1	RO	RW	RW	n/a	TPIDRURO
EL3 not implemented	x	1	1	RO	n/a	RW	n/a	TPIDRURO
EL3 using AArch64	x	x	0	RO	RW	n/a	n/a	TPIDRURO
EL3 using AArch64	x	0	1	RO	RW	RW	n/a	TPIDRURO
EL3 using AArch64	x	1	1	RO	n/a	RW	n/a	TPIDRURO

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.T13==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.T13==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 [HSTR.T13==1](#), Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# TPIDRURW, PL0 Read/Write Software Thread ID Register

The TPIDRURW characteristics are:

## Purpose

Provides a location where software executing at EL0 can store thread identifying information, for OS management purposes.

The PE makes no use of this register.

This register is part of the Thread and process ID registers functional group.

## Configuration

AArch32 System register TPIDRURW is architecturally mapped to AArch64 System register [TPIDR\\_EL0\[31:0\]](#).

The PE never updates this register. This means the register is always UNKNOWN on reset.

## Attributes

TPIDRURW is a 32-bit register.

## Field descriptions

The TPIDRURW 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
Thread ID																															

### Bits [31:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

## Accessing the TPIDRURW

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>, c13, c0, 2	000	010	1101	1111	0000

## 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	TPIDRURW_s
EL3 using AArch32	x	0	1	RW	RW	RW	RW	TPIDRURW_ns

EL3 using AArch32	x	1	1	RW	n/a	RW	RW	TPIDRURW_ns
EL3 not implemented	x	x	0	RW	RW	n/a	n/a	TPIDRURW
EL3 not implemented	x	0	1	RW	RW	RW	n/a	TPIDRURW
EL3 not implemented	x	1	1	RW	n/a	RW	n/a	TPIDRURW
EL3 using AArch64	x	x	0	RW	RW	n/a	n/a	TPIDRURW
EL3 using AArch64	x	0	1	RW	RW	RW	n/a	TPIDRURW
EL3 using AArch64	x	1	1	RW	n/a	RW	n/a	TPIDRURW

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<sup>®</sup> 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](#).T13==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](#).T13==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 [HSTR](#).T13==1, Non-secure accesses to this register from EL0 and EL1 are trapped to Hyp mode.

02/05/2017 15:43

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



# TTBCR, Translation Table Base Control Register

The TTBCR characteristics are:

## Purpose

The control register for stage 1 of the PL1&0 translation regime. Its controls include:

- Where the VA range is split between addresses translated using [TTBR0](#) and addresses translated using [TTBR1](#).
- The translation table format used by this stage of translation.

In ARMv8.2, when the value of TTBCR.{EAE, T2E} is {1, 1}, TTBCR is used with [TTBCR2](#).

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch32 System register TTBCR is architecturally mapped to AArch64 System register [TCR\\_EL1\[31:0\]](#).

The current translation table format determines which format of the register is used.

When EL3 is using AArch32, write access to TTBCR(S) is disabled when the CP15SDISABLE signal is asserted HIGH.

Some 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 then:

- The EAE bit resets to 0 in both the Secure and the Non-secure instances of the register.
- Other reset values apply only to the Secure instance of the register.

## Attributes

TTBCR is a 32-bit register.

## Field descriptions

The TTBCR bit assignments are:

### For all register layouts:

#### EAE, bit [31]

Extended Address Enable. The meanings of the possible values of this bit are:

EAE	Meaning
0	Use the VMSAv8-32 translation system with the Short-descriptor translation table format.
1	Use the VMSAv8-32 translation system with the Long-descriptor translation table format.

### When TTBCR.EAE==0:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<a href="#">EAE</a>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<a href="#">PD1</a>	<a href="#">PD0</a>	0			<a href="#">N</a>	

#### EAE, bit [31]

Extended Address Enable. The meanings of the possible values of this bit are:

EAE	Meaning
0	Use the VMSAv8-32 translation system with the Short-descriptor translation table format.

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

### Bits [30:6]

Reserved, RES0.

### PD1, bit [5]

Translation table walk disable for translations using [TTBR1](#). This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using [TTBR1](#). The encoding of this bit is:

PD1	Meaning
0	Perform translation table walks using <a href="#">TTBR1</a> .
1	A TLB miss on an address that is translated using <a href="#">TTBR1</a> generates a Translation fault. No translation table walk is performed.

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

### PD0, bit [4]

Translation table walk disable for translations using [TTBR0](#). This bit controls whether a translation table walk is performed on a TLB miss for an address that is translated using [TTBR0](#). The encoding of this bit is:

PD0	Meaning
0	Perform translation table walks using <a href="#">TTBR0</a> .
1	A TLB miss on an address that is translated using <a href="#">TTBR0</a> generates a Translation fault. No translation table walk is performed.

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

### Bit [3]

Reserved, RES0.

### N, bits [2:0]

Indicate the width of the base address held in [TTBR0](#). In [TTBR0](#), the base address field is bits[31:14-N]. The value of N also determines:

- Whether [TTBR0](#) or [TTBR1](#) is used as the base address for translation table walks.
- The size of the translation table pointed to by [TTBR0](#).

N can take any value from 0 to 7, that is, from 0b000 to 0b111.

When N has its reset value of 0, the translation table base is compatible with ARMv5 and ARMv6.

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

## When TTBCR.EAE==1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EAE	IMP DEF	SH1	ORGN1	IRGN1	EPD1	A1	0	0	0	T1SZ	0	0	SH0	ORGN0	IRGN0	EPD0	T2E	0	0	0	T0SZ										

### EAE, bit [31]

Extended Address Enable. The meanings of the possible values of this bit are:

EAE	Meaning
1	Use the VMSAv8-32 translation system with the Long-descriptor translation table format.

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

#### IMP DEF, bit [30]

IMPLEMENTATION DEFINED.

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

#### SH1, bits [29:28]

Shareability attribute for memory associated with translation table walks using [TTBR1](#). 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 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

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

#### ORGN1, bits [27:26]

Outer cacheability attribute for memory associated with translation table walks using [TTBR1](#).

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

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

#### IRGN1, bits [25:24]

Inner cacheability attribute for memory associated with translation table walks using [TTBR1](#).

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

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

#### EPD1, bit [23]

Translation table walk disable for translations using [TTBR1](#). This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using [TTBR1](#). The encoding of this bit is:

EPD1	Meaning
0	Perform translation table walks using <a href="#">TTBR1</a> .
1	A TLB miss on an address that is translated using <a href="#">TTBR1</a> generates a Translation fault. No translation table walk is performed.

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

**A1, bit [22]**

Selects whether [TTBR0](#) or [TTBR1](#) defines the ASID. The encoding of this bit is:

A1	Meaning
0	<a href="#">TTBR0</a> .ASID defines the ASID.
1	<a href="#">TTBR1</a> .ASID defines the ASID.

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

**Bits [21:19]**

Reserved, RES0.

**T1SZ, bits [18:16]**

See 'Selecting between TTBR0 and TTBR1, VMSAv8-32 Long-descriptor translation table format' in the ARMv8 ARM for how [TTBCR](#).{T1SZ, T0SZ} determine the input address ranges and memory region sizes translated using [TTBR0](#) and [TTBR1](#).

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

**Bits [15:14]**

Reserved, RES0.

**SH0, bits [13:12]**

Shareability attribute for memory associated with translation table walks using [TTBR0](#).

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 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

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

**ORGN0, bits [11:10]**

Outer cacheability attribute for memory associated with translation table walks using [TTBR0](#).

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

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

**IRGN0, bits [9:8]**

Inner cacheability attribute for memory associated with translation table walks using [TTBR0](#).

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

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

### EPD0, bit [7]

Translation table walk disable for translations using [TTBR0](#). This bit controls whether a translation table walk is performed on a TLB miss, for an address that is translated using [TTBR0](#). The encoding of this bit is:

EPD0	Meaning
0	Perform translation table walks using <a href="#">TTBR0</a> .
1	A TLB miss on an address that is translated using <a href="#">TTBR0</a> generates a Translation fault. No translation table walk is performed.

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

### T2E, bit [6]

In ARMv8.2:

TTBCR2 Enable.

Defined values are:

T2E	Meaning
0	<a href="#">TTBCR2</a> is disabled. The contents of <a href="#">TTBCR2</a> are treated as 0 for all purposes other than reading or writing the register.
1	<a href="#">TTBCR2</a> is enabled.

If TTBCR.EAE==0, then the behavior is as if the bit is 0.

This bit is RES0 if ARMv8.2-AA32HPD is not implemented.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

### Bits [5:3]

Reserved, RES0.

### T0SZ, bits [2:0]

See 'Selecting between TTBR0 and TTBR1, VMSAv8-32 Long-descriptor translation table format' in the ARMv8 ARM for how [TTBCR](#).{T1SZ, T0SZ} determine the input address ranges and memory region sizes translated using [TTBR0](#) and [TTBR1](#).

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

## Accessing the TTBCR

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>, c2, c0, 2	000	010	0010	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	TTBCR
EL3 not implemented	x	0	1	-	RW	RW	n/a	TTBCR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	TTBCR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	TTBCR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	TTBCR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	TTBCR
EL3 using AArch32	x	0	1	-	RW	RW	RW	TTBCR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	TTBCR_ns
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	TTBCR_s

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

When EL3 is using AArch32, write access to TTBCR\_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<sup>®</sup> 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.T2](#)==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.T2](#)==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.T2](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# TTBCR2, Translation Table Base Control Register 2

The TTBCR2 characteristics are:

## Purpose

The second control register for stage 1 of the PL1&0 translation regime.

If ARMv8.2-AA32HPD is not implemented and ARMv8.2-TTPBHA is not implemented then this register is not implemented and its encoding is unallocated. Otherwise:

- When the value of [TTBCR](#).{EAE, T2E} is not {1, 1} the contents of TTBCR2 are treated as zero for all purposes other than reading or writing the register.
- When the value of [TTBCR](#).{EAE, T2E} is {1, 1} TTBCR2 is used with [TTBCR](#).

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch32 System register TTBCR2 is architecturally mapped to AArch64 System register [TCR\\_EL1\[63:32\]](#).

When EL3 is using AArch32, write access to TTBCR2(S) is disabled when the CP15SDISABLE signal is asserted HIGH.

RW fields in this register reset to architecturally UNKNOWN values.

This register is introduced in ARMv8.2.

## Attributes

TTBCR2 is a 32-bit register.

## Field descriptions

The TTBCR2 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	HWU162	HWU161	HWU160	HWU159	HWU062	HWU061	HWU060	HWU059	HPD1	HPD0	0	0	0	0	0	0	0	0

### Bits [31:19]

Reserved, RES0.

### HWU162, bit [18]

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR1](#) if TTBCR2.HPD1==1 and [TTBCR](#).T2E==1.

Defined values are:

HWU162	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TTBCR2.HPD1 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**HWU161, bit [17]**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR1](#) if TTBCR2.HPD1==1 and [TTBCR.T2E](#)==1.

Defined values are:

HWU161	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TTBCR2.HPD1 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**HWU160, bit [16]**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR1](#) if TTBCR2.HPD1==1 and [TTBCR.T2E](#)==1.

Defined values are:

HWU160	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TTBCR2.HPD1 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**HWU159, bit [15]**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR1](#) if TTBCR2.HPD1==1 and [TTBCR.T2E](#)==1.

Defined values are:

HWU159	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TTBCR2.HPD1 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**HWU062, bit [14]**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR0](#) if TTBCR2.HPD0==1 and [TTBCR.T2E](#)==1.

Defined values are:

HWU062	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TTBCR2.HPD0 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.



**HWU061, bit [13]**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR0](#) if TTBCR2.HPD0==1 and [TTBCR.T2E](#)==1.

Defined values are:

HWU061	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TTBCR2.HPD0 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**HWU060, bit [12]**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR0](#) if TTBCR2.HPD0==1 and [TTBCR.T2E](#)==1.

Defined values are:

HWU060	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TTBCR2.HPD0 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**HWU059, bit [11]**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR0](#) if TTBCR2.HPD0==1 and [TTBCR.T2E](#)==1.

Defined values are:

HWU059	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TTBCR2.HPD0 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**HPD1, bit [10]**

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, XNTable, and PXNTable, in the translation tables pointed to by [TTBR1](#).

Defined values are:

HPD1	Meaning
0	Hierarchical permissions are enabled.
1	Hierarchical permissions are disabled if <a href="#">TTBCR.T2E</a> == 1.

When disabled, the permissions are treated as if the bits are 0.

If [TTBCR.T2E](#) == 0, the hierarchical permissions are enabled.

This bit is RES0 if ARMv8.2-AA32HPD is not implemented.

**HPD0, bit [9]**

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, XNTable, and PXNTable, in the translation tables pointed to by [TTBR0](#).

Defined values are:

HPD0	Meaning
0	Hierarchical permissions are enabled.
1	Hierarchical permissions are disabled if <a href="#">TTBCR.T2E</a> == 1.

When disabled, the permissions are treated as if the bits are 0.

If [TTBCR.T2E](#) == 0, the hierarchical permissions are enabled.

This bit is RES0 if ARMv8.2-AA32HPD is not implemented.

**Bits [8:0]**

Reserved, RES0.

**Accessing the TTBCR2**

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>, c2, c0, 3	000	011	0010	1111	0000

**Accessibility**

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	
EL3 using AArch32	x	0	1	-	RW	RW	RW	TTBCR2_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	TTBCR2_ns
EL3 not implemented	x	x	0	-	RW	n/a	n/a	TTBCR2
EL3 not implemented	x	0	1	-	RW	RW	n/a	TTBCR2
EL3 not implemented	x	1	1	-	n/a	RW	n/a	TTBCR2
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	TTBCR2
EL3 using AArch64	x	0	1	-	RW	RW	n/a	TTBCR2
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	TTBCR2
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	TTBCR2_s

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

When EL3 is using AArch32, write access to TTBCR2\_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<sup>®</sup> 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](#).T2==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](#).T2==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](#).T2==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# TTBR0, Translation Table Base Register 0

The TTBR0 characteristics are:

## Purpose

Holds the base address of the translation table for the initial lookup for stage 1 of the translation of an address from the lower VA range in the PL1&0 translation regime, and other information for this translation regime.

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch32 System register TTBR0 is architecturally mapped to AArch64 System register [TTBR0\\_EL1](#).

[TTBCR](#).EAE determines which TTBR0 format is used:

**EAE==0**

32-bit format is used. TTBR0[63:32] are ignored.

**EAE==1**

64-bit format is used.

When EL3 is using AArch32, write access to TTBR0(S) is disabled when the CP15SSDISABLE signal is asserted HIGH.

Used in conjunction with the [TTBCR](#). When the 64-bit TTBR0 format is used, cacheability and shareability information is held in the [TTBCR](#), not in TTBR0.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

TTBR0 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 TTBR0 bit assignments are:

### When TTBCR.EAE==0:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTB0														IRGN[0]		NOS	RGN	IMP	S	IRGN[1]											

### TTB0, bits [31:7]

Translation table base address, bits[31:x], where x is 14-(TTBCR.N). Register bits [x-1:7] are RES0, with the additional requirement that if these bits are not all zero, this is a misaligned translation table base address, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Register bits [x-1:7] are treated as if all the bits are zero. The value read back from these bits is either the value written or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

### IRGN[0], bit [6]

See the IRGN[1] description.

**NOS, bit [5]**

Not Outer Shareable. When the value of TTBR0.S is 1, indicates whether the memory associated with a translation table walk is Inner Shareable or Outer Shareable:

NOS	Meaning
0	Memory is Outer Shareable.
1	Memory is Inner Shareable.

This bit is ignored when the value of TTBR0.S is 0.

**RGN, bits [4:3]**

Region bits. Indicates the Outer cacheability attributes for the memory associated with the translation table walks:

RGN	Meaning
00	Normal memory, Outer Non-cacheable.
01	Normal memory, Outer Write-Back Write-Allocate Cacheable.
10	Normal memory, Outer Write-Through Cacheable.
11	Normal memory, Outer Write-Back no Write-Allocate Cacheable.

**IMP, bit [2]**

The effect of this bit is IMPLEMENTATION DEFINED. If the translation table implementation does not include any IMPLEMENTATION DEFINED features this bit is UNK/SBZP.

**S, bit [1]**

Shareable. Indicates whether the memory associated with the translation table walks is Non-shareable:

S	Meaning
0	Memory is Non-shareable.
1	Memory is shareable. The TTBR0.NOS field indicates whether the memory is Inner Shareable or Outer Shareable.

**IRGN[1], bit [0]**

Inner region bits. IRGN[1:0] indicate the Inner Cacheability attributes for the memory associated with the translation table walks. The possible values of IRGN[1:0] are:

IRGN	Meaning
00	Normal memory, Inner Non-cacheable.
01	Normal memory, Inner Write-Back Write-Allocate Cacheable.
10	Normal memory, Inner Write-Through Cacheable.
11	Normal memory, Inner Write-Back no Write-Allocate Cacheable.

**Note**

The encoding of the IRGN bits is counter-intuitive, with register bit[6] being IRGN[0] and register bit[0] being IRGN[1]. This encoding is chosen to give a consistent encoding of memory region types and to ensure that software written for ARMv7 without the Multiprocessing Extensions can run unmodified on an implementation that includes the functionality introduced by the ARMv7 Multiprocessing Extensions.

**When TTBCR.EAE==1:**

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

ASID

BADDR

BADDR

CnP

**Bits [63:56]**

Reserved, RES0.

**ASID, bits [55:48]**

An ASID for the translation table base address. The [TTBCR.A1](#) field selects either TTBR0.ASID or TTBR1.ASID.

**BADDR, bits [47:1]**

Translation table base address, bits[47:x], Bits [x-1:1] are RES0, with the additional requirement that if bits[x-1:3] are not all zero, this is a misaligned translation table base address, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Register bits [x-1:3] are treated as if all the bits are zero. The value read back from these bits is either the value written or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

x is determined from the value of [TTBCR.T0SZ](#) as follows:

- If [TTBCR.T0SZ](#) is 0 or 1,  $x = 5 - \text{TTBCR.T0SZ}$ .
- If [TTBCR.T0SZ](#) is greater than 1,  $x = 14 - \text{TTBCR.T0SZ}$ .

If bits[47:40] of the translation table base address are not zero, an Address size fault is generated.

**CnP, bit [0]****In ARMv8.2:**

Common not Private. In an implementation that includes ARMv8.2-TTCNP, when [TTBCR.EAE](#) == 1, indicates whether each entry that is pointed to by TTBR0 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR0.CnP is 1.

CnP	Meaning
0	The translation table entries pointed to by this instance of TTBR0, for the current ASID, are permitted to differ from corresponding entries for this instance of TTBR0 for other PEs in the Inner Shareable domain. This is not affected by: <ul style="list-style-type: none"> <li>• The value of TTBR0.CnP on those other PEs.</li> <li>• The value of <a href="#">TTBCR.EAE</a> on those other PEs.</li> <li>• The value of the current ASID or, for the Non-secure instance of TTBR0, the value of the current VMID.</li> </ul>
1	The translation table entries pointed to by this instance of TTBR0 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR0.CnP is 1 for this instance of TTBR0 and all of the following apply: <ul style="list-style-type: none"> <li>• The translation table entries are pointed to by this instance of TTBR0.</li> <li>• The value of the applicable <a href="#">TTBCR.EAE</a> field is 1.</li> <li>• The ASID is the same as the current ASID.</li> <li>• For the Non-secure instance of TTBR0, the VMID is the same as the current VMID.</li> </ul>

When a TLB combines entries from stage 1 translation and stage 2 translation into a single entry, that entry can only be shared between different PEs if the value of the CnP bit is 1 for both stage 1 and stage 2.

**Note**

If the value of the TTBR0.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR0s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONSTRAINED UNPREDICTABLE, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values' in the ARMv8-A ARM appendix K1.

In an implementation that does not include ARMv8.2-TTCNP this field is RES0.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

## Accessing the TTBR0

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>, c2, c0, 0	000	000	0010	1111	0000

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>, c2	0000	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	n/a	n/a	TTBR0
EL3 not implemented	x	0	1	-	RW	RW	n/a	TTBR0
EL3 not implemented	x	1	1	-	n/a	RW	n/a	TTBR0
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	TTBR0
EL3 using AArch64	x	0	1	-	RW	RW	n/a	TTBR0
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	TTBR0
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	TTBR0_s
EL3 using AArch32	x	0	1	-	RW	RW	RW	TTBR0_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	TTBR0_ns

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

When EL3 is using AArch32, write access to TTBR0\_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.T2==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](#).T2==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](#).T2==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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



# TTBR1, Translation Table Base Register 1

The TTBR1 characteristics are:

## Purpose

Holds the base address of the translation table for the initial lookup for stage 1 of the translation of an address from the higher VA range in the PL1&0 translation regime, and other information for this translation regime.

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch32 System register TTBR1 is architecturally mapped to AArch64 System register [TTBR1\\_EL1](#).

[TTBCR](#).EAE determines which TTBR1 format is used:

**EAE==0**

32-bit format is used. TTBR1[63:32] are ignored.

**EAE==1**

64-bit format is used.

Used in conjunction with the [TTBCR](#). When the 64-bit TTBR1 format is used, cacheability and shareability information is held in the [TTBCR](#), not in TTBR1.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

TTBR1 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 TTBR1 bit assignments are:

### When TTBCR.EAE==0:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
												TTB1												IRGN[0]		NOS	RGN	IMP	S	IRGN[1]						

### TTB1, bits [31:7]

Translation table base address, bits[31:14]. Register bits [13:7] are RES0, with the additional requirement that if these bits are not all zero, this is a misaligned translation table base address, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Register bits [13:7] are treated as if all the bits are zero. The value read back from these bits is either the value written or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

### IRGN[0], bit [6]

See the [IRGN\[1\]](#) description.

### NOS, bit [5]

Not Outer Shareable. When the value of TTBR1.S is 1, indicates whether the memory associated with a translation table walk is Inner Shareable or Outer Shareable:

NOS	Meaning
0	Memory is Outer Shareable.
1	Memory is Inner Shareable.

This bit is ignored when the value of TTBR1.S is 0.

#### RGN, bits [4:3]

Region bits. Indicates the Outer cacheability attributes for the memory associated with the translation table walks:

RGN	Meaning
00	Normal memory, Outer Non-cacheable.
01	Normal memory, Outer Write-Back Write-Allocate Cacheable.
10	Normal memory, Outer Write-Through Cacheable.
11	Normal memory, Outer Write-Back no Write-Allocate Cacheable.

#### IMP, bit [2]

The effect of this bit is IMPLEMENTATION DEFINED. If the translation table implementation does not include any IMPLEMENTATION DEFINED features this bit is UNK/SBZP.

#### S, bit [1]

Shareable. Indicates whether the memory associated with the translation table walks is Non-shareable:

S	Meaning
0	Memory is Non-shareable.
1	Memory is shareable. The TTBR1.NOS field indicates whether the memory is Inner Shareable or Outer Shareable.

#### IRGN[1], bit [0]

Inner region bits. IRGN[1:0] indicate the Inner Cacheability attributes for the memory associated with the translation table walks. The possible values of IRGN[1:0] are:

IRGN	Meaning
00	Normal memory, Inner Non-cacheable.
01	Normal memory, Inner Write-Back Write-Allocate Cacheable.
10	Normal memory, Inner Write-Through Cacheable.
11	Normal memory, Inner Write-Back no Write-Allocate Cacheable.

#### Note

The encoding of the IRGN bits is counter-intuitive, with register bit[6] being IRGN[0] and register bit[0] being IRGN[1]. This encoding is chosen to give a consistent encoding of memory region types and to ensure that software written for ARMv7 without the Multiprocessing Extensions can run unmodified on an implementation that includes the functionality introduced by the ARMv7 Multiprocessing Extensions.

#### When TTBCR.EAE==1:

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

#### Bits [63:56]

Reserved, RES0.

**ASID, bits [55:48]**

An ASID for the translation table base address. The [TTBCR.A1](#) field selects either TTBR0.ASID or TTBR1.ASID.

**BADDR, bits [47:1]**

Translation table base address, bits[47:x], Bits [x-1:1] are RES0, with the additional requirement that if bits[x-1:3] are not all zero, this is a misaligned translation table base address, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Register bits [x-1:3] are treated as if all the bits are zero. The value read back from these bits is either the value written or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

x is determined from the value of [TTBCR.T1SZ](#) as follows:

- If [TTBCR.T1SZ](#) is 0 or 1,  $x = 5 - \text{TTBCR.T1SZ}$ .
- If [TTBCR.T1SZ](#) is greater than 1,  $x = 14 - \text{TTBCR.T1SZ}$ .

If bits[47:40] of the translation table base address are not zero, an Address size fault is generated.

**CnP, bit [0]**

In ARMv8.2:

Common not Private. In an implementation that includes ARMv8.2-TTCNP, when [TTBCR.EAE](#) == 1, indicates whether each entry that is pointed to by TTBR1 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR1.CnP is 1.

CnP	Meaning
0	The translation table entries pointed to by this instance of TTBR1, for the current ASID, are permitted to differ from corresponding entries for this instance of TTBR1 for other PEs in the Inner Shareable domain. This is not affected by: <ul style="list-style-type: none"> <li>• The value of TTBR1.CnP on those other PEs.</li> <li>• The value of <a href="#">TTBCR.EAE</a> on those other PEs.</li> <li>• The value of the current ASID or, for the Non-secure instance of TTBR1, the value of the current VMID.</li> </ul>
1	The translation table entries pointed to by this instance of TTBR1 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR1.CnP is 1 for this instance of TTBR1 and all of the following apply: <ul style="list-style-type: none"> <li>• The translation table entries are pointed to by this instance of TTBR1.</li> <li>• The value of the applicable <a href="#">TTBCR.EAE</a> field is 1.</li> <li>• The ASID is the same as the current ASID.</li> <li>• For the Non-secure instance of TTBR1, the VMID is the same as the current VMID.</li> </ul>

When a TLB combines entries from stage 1 translation and stage 2 translation into a single entry, that entry can only be shared between different PEs if the value of the CnP bit is 1 for both stage 1 and stage 2.

**Note**

If the value of the TTBR1.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR1s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONSTRAINED UNPREDICTABLE, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values' in the ARMv8-A ARM appendix K1.

In an implementation that does not include ARMv8.2-TTCNP this field is RES0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

## Accessing the TTBR1

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>, c2, c0, 1	000	001	0010	1111	0000

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, 1, <Rt>, <Rt2>, c2	0001	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	n/a	n/a	TTBR1
EL3 not implemented	x	0	1	-	RW	RW	n/a	TTBR1
EL3 not implemented	x	1	1	-	n/a	RW	n/a	TTBR1
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	TTBR1
EL3 using AArch64	x	0	1	-	RW	RW	n/a	TTBR1
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	TTBR1
EL3 using AArch32	x	0	1	-	RW	RW	RW	TTBR1_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	TTBR1_ns
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	TTBR1_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<sup>®</sup> 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.T2==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](#).T2==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](#).T2==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# VBAR, Vector Base Address Register

The VBAR characteristics are:

## Purpose

When high exception vectors are not selected, holds the vector base address for exceptions that are not taken to Monitor mode or to Hyp mode.

Software must program VBAR(NS) with the required initial value as part of the PE boot sequence.

This register is part of the Exception and fault handling registers functional group.

## Configuration

AArch32 System register VBAR is architecturally mapped to AArch64 System register [VBAR\\_EL1\[31:0\]](#).

When EL3 is using AArch32, write access to VBAR(S) is disabled when the CP15SDISABLE signal is asserted HIGH.

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

VBAR is a 32-bit register.

## Field descriptions

The VBAR 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	
Vector Base Address																												0	0	0	0	0

### Bits [31:5]

Vector Base Address. Bits[31:5] of the base address of the exception vectors for exceptions taken to this Exception level. Bits[4:0] of an exception vector are the exception offset.

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

### Bits [4:0]

Reserved, RES0.

## Accessing the VBAR

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, c0, 0	000	000	1100	1111	0000

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility				Instance
	E2H	TGE	NS	EL0	EL1	EL2	EL3	
EL3 using AArch32	x	x	0	-	n/a	n/a	RW	VBAR_s
EL3 using AArch32	x	0	1	-	RW	RW	RW	VBAR_ns
EL3 using AArch32	x	1	1	-	n/a	RW	RW	VBAR_ns
EL3 not implemented	x	x	0	-	RW	n/a	n/a	VBAR
EL3 not implemented	x	0	1	-	RW	RW	n/a	VBAR
EL3 not implemented	x	1	1	-	n/a	RW	n/a	VBAR
EL3 using AArch64	x	x	0	-	RW	n/a	n/a	VBAR
EL3 using AArch64	x	0	1	-	RW	RW	n/a	VBAR
EL3 using AArch64	x	1	1	-	n/a	RW	n/a	VBAR

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

When EL3 is using AArch32, write access to VBAR\_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 [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.E2H==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.

# VMPIDR, Virtualization Multiprocessor ID Register

The VMPIDR characteristics are:

## Purpose

Holds the value of the Virtualization Multiprocessor ID. This is the value returned by Non-secure EL1 reads of [MPIDR](#).

This register is part of:

- The Identification registers functional group.
- The Virtualization registers functional group.

## Configuration

AArch32 System register VMPIDR is architecturally mapped to AArch64 System register [VMPIDR\\_EL2\[31:0\]](#).

If EL2 is not implemented but EL3 is implemented, this register takes the value of the [MPIDR](#).

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

VMPIDR is a 32-bit register.

## Field descriptions

The VMPIDR 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
<a href="#">M</a>	<a href="#">U</a>	0	0	0	0	0	<a href="#">MT</a>	<a href="#">Aff2</a>								<a href="#">Aff1</a>								<a href="#">Aff0</a>							

### M, bit [31]

Indicates whether this implementation includes the functionality introduced by the ARMv7 Multiprocessing Extensions. The possible values of this bit are:

M	Meaning
0	This implementation does not include the ARMv7 Multiprocessing Extensions functionality.
1	This implementation includes the ARMv7 Multiprocessing Extensions functionality.

In ARMv8 this bit is RES1.

### U, bit [30]

Indicates a Uniprocessor system, as distinct from PE 0 in a multiprocessor system. The possible values of this bit are:

U	Meaning
0	Processor is part of a multiprocessor system.
1	Processor is part of a uniprocessor system.

When this register has an architecturally-defined reset value, this field resets to the value of [MPIDR](#).U.

### Bits [29:25]

Reserved, RES0.



**MT, bit [24]**

Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. The possible values of this bit are:

MT	Meaning
0	Performance of PEs at the lowest affinity level is largely independent.
1	Performance of PEs at the lowest affinity level is very interdependent.

When this register has an architecturally-defined reset value, this field resets to the value of [MPIDR](#).MT.

**Aff2, bits [23:16]**

Affinity level 2. The least significant affinity level field, for this PE in the system.

When this register has an architecturally-defined reset value, this field resets to the value of [MPIDR](#).Aff2.

**Aff1, bits [15:8]**

Affinity level 1. The intermediate affinity level field, for this PE in the system.

When this register has an architecturally-defined reset value, this field resets to the value of [MPIDR](#).Aff1.

**Aff0, bits [7:0]**

Affinity level 0. The most significant affinity level field, for this PE in the system.

When this register has an architecturally-defined reset value, this field resets to the value of [MPIDR](#).Aff0.

## Accessing the VMPIDR

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, 5	100	101	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<sup>®</sup> 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  :

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

02/05/2017 15:43

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

# 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 'Identification registers, functional group' in the ARMv8 ARM, section G4.18.1.

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<sup>®</sup> 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  :

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

02/05/2017 15:43

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

# VTCR, Virtualization Translation Control Register

The VTCR characteristics are:

## Purpose

The control register for stage 2 of the Non-secure PL1&0 translation regime.

### Note

This stage of translation always uses the Long-descriptor translation table format.

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

## Configuration

AArch32 System register VTCR is architecturally mapped to AArch64 System register [VTCR\\_EL2](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

VTCR is a 32-bit register.

## Field descriptions

The VTCR 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
1	0	0	<a href="#">HWU62</a>	<a href="#">HWU61</a>	<a href="#">HWU60</a>	<a href="#">HWU59</a>	0	0	0	0	0	0	0	0	0	0	0	<a href="#">SH0</a>	<a href="#">ORGN0</a>	<a href="#">IRGN0</a>	<a href="#">SL0</a>	0	<a href="#">S</a>						<a href="#">T0SZ</a>		

### Bit [31]

Reserved, RES1.

### Bits [30:29]

Reserved, RES0.

### HWU62, bit [28]

In ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 2 translation table block or level 3 entry.

Defined values are:

HWU62	Meaning
0	The stage 2 translation table entry block or level 3 entry cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 2 translation table entry block or level 3 entry can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

#### HWU61, bit [27]

In ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 2 translation table block or level 3 entry.

Defined values are:

HWU61	Meaning
0	The stage 2 translation table entry block or level 3 entry cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 2 translation table entry block or level 3 entry can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

#### HWU60, bit [26]

In ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 2 translation table block or level 3 entry.

Defined values are:

HWU60	Meaning
0	The stage 2 translation table entry block or level 3 entry cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 2 translation table entry block or level 3 entry can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

#### HWU59, bit [25]

In ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 2 translation table block or level 3 entry.

Defined values are:

HWU59	Meaning
0	The stage 2 translation table entry block or level 3 entry cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 2 translation table entry block or level 3 entry can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**Bits [24:14]**

Reserved, RES0.

**SH0, bits [13:12]**Shareability attribute for memory associated with translation table walks using [VTTBR](#).

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 'Unallocated values in fields of AArch32 System registers and translation table entries' in the ARM ARM, section K1.1.11.

**ORGN0, bits [11:10]**Outer cacheability attribute for memory associated with translation table walks using [VTTBR](#).

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 [VTTBR](#).

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

**SL0, bits [7:6]**Starting level for translation table walks using [VTTBR](#).

SL0	Meaning
00	Start at level 2
01	Start at level 1

All other values are reserved. If this field is programmed to a reserved value, or to a value that is not consistent with the programming of T0SZ, then a stage 2 level 1 Translation fault is generated.

**Bit [5]**

Reserved, RES0.

**S, bit [4]**

Sign extension bit. This bit must be programmed to the value of T0SZ[3]. If it is not, then the behavior is CONSTRAINED UNPREDICTABLE and the stage 2 T0SZ value is treated as an UNKNOWN value, see 'Misprogramming VTCR.S' in the ARM ARM.



**T0SZ, bits [3:0]**

The size offset of the memory region addressed by [VTTBR](#). The region size is  $2^{(32-T0SZ)}$  bytes.

This field holds a four-bit signed integer value, meaning it supports values from -8 to 7.

**Note**

This is different from the other translation control registers, where TnSZ holds a three-bit unsigned integer, supporting values from 0 to 7.

If this field is programmed to a value that is not consistent with the programming of SL0 then a stage 2 level 1 Translation fault is generated.

**Accessing the VTCR**

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>, c2, c1, 2	100	010	0010	1111	0001

**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<sup>®</sup> 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.T2](#)==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.T2](#)==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.T2](#)==1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.



# VTTBR, Virtualization Translation Table Base Register

The VTTBR characteristics are:

## Purpose

Holds the base address of the translation table for the initial lookup for stage 2 of an address translation in the Non-secure PL1&0 translation regime, and other information for this translation regime.

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

## Configuration

AArch32 System register VTTBR is architecturally mapped to AArch64 System register [VTTBR\\_EL2](#).

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 with EL2 using AArch32, or into EL3 with EL3 using AArch32. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

VTTBR is a 64-bit register.

## Field descriptions

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

### Bits [63:56]

Reserved, RES0.

### VMID, bits [55:48]

The VMID for the translation table.

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

### BADDR, bits [47:1]

Translation table base address, bits[47:x], Bits [x-1:1] are RES0, with the additional requirement that if bits[x-1:3] are not all zero, this is a misaligned translation table base address, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Register bits [x-1:3] are treated as if all the bits are zero. The value read back from these bits is either the value written or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

x is determined from the value of [VTCR.SL0](#) and [VTCR.T0SZ](#) as follows:

- If [VTCR.SL0](#) is 00, meaning that lookup starts at level 2, then x is 14 - [VTCR.T0SZ](#).
- If [VTCR.SL0](#) is 01, meaning that lookup starts at level 1, then x is 5 - [VTCR.T0SZ](#).

- If [VTCR.SL0](#) is either 10 or 11 then a stage 2 level 1 Translation fault is generated.

If bits[47:40] of the translation table base address are not zero, an Address size fault is generated.

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

### CnP, bit [0]

In ARMv8.2:

Common not Private. In an implementation that includes ARMv8.2-TTCNP, indicates whether each entry that is pointed to by VTTBR is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of VTTBR.CnP is 1.

CnP	Meaning
0	The translation table entries pointed to by VTTBR are permitted to differ from the entries for VTTBR for other PEs in the Inner Shareable domain. This is not affected by the value of the current VMID.
1	The translation table entries pointed to by VTTBR are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of VTTBR.CnP is 1 and the VMID is the same as the current VMID.

When a TLB combines entries from stage 1 translation and stage 2 translation into a single entry, that entry can only be shared between different PEs if the value of the CnP bit is 1 for both stage 1 and stage 2.

### Note

If the value of the VTTBR.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those VTTBRs do not point to the same translation table entries when the VMID value is the same as the current VMID, then the results of translations are CONSTRAINED UNPREDICTABLE, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values' in the ARMv8-A ARM appendix K1.

In an implementation that does not include ARMv8.2-TTCNP this field is RES0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

## Accessing the VTTBR

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>, c2	0110	1111	0010

## 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<sup>®</sup> 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T2](#)=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 [HSTR\\_EL2.T2](#)=1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T2](#)=1, Non-secure accesses to this register from EL1 are trapped to Hyp mode.

02/05/2017 15:43

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

# 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

[TLBIMVAAL](#): TLB Invalidate by VA, All ASID, Last level

[TLBIMVAALIS](#): TLB Invalidate by VA, All ASID, Last level, Inner Shareable

[TLBIMVAH](#): TLB Invalidate by VA, Hyp mode

[TLBIMVAHIS](#): TLB Invalidate by VA, Hyp mode, Inner Shareable

[TLBIMVAIS](#): TLB Invalidate by VA, Inner Shareable

[TLBIMVAL](#): TLB Invalidate by VA, Last level

[TLBIMVALH](#): TLB Invalidate by VA, Last level, Hyp mode

[TLBIMVALHIS](#): TLB Invalidate by VA, Last level, Hyp mode, Inner Shareable

[TLBIMVALIS](#): TLB Invalidate by VA, Last level, Inner Shareable

02/05/2017 15:43

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

# ATS12NSOPR, Address Translate Stages 1 and 2 Non-secure Only PL1 Read

The ATS12NSOPR characteristics are:

## Purpose

Performs stage 1 and 2 address translations as defined for PL1 and the Non-secure state, with permissions as if reading from the given virtual address.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

ATS12NSOPR is a 32-bit System instruction.

## Field descriptions

The ATS12NSOPR input value 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
Input address for translation																															

### Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This instruction takes a VA as input. The resulting address is the PA that is the output address of the stage 2 translation.

## Executing the ATS12NSOPR instruction

This instruction is executed 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>, c7, c8, 4	000	100	0111	1111	1000

## Accessibility

The instruction is executable as follows:

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



This table applies to all syntax that can be used to execute this instruction.

If EL3 is implemented and is using AArch64, execution of ATS12NSOPR in 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<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T7](#)==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 [HSTR\\_EL2.T7](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T7](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ATS12NSOPW, Address Translate Stages 1 and 2 Non-secure Only PL1 Write

The ATS12NSOPW characteristics are:

## Purpose

Performs stage 1 and 2 address translations as defined for PL1 and the Non-secure state, with permissions as if writing to the given virtual address.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

ATS12NSOPW is a 32-bit System instruction.

## Field descriptions

The ATS12NSOPW input value 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
Input address for translation																															

### Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This instruction takes a VA as input. The resulting address is the PA that is the output address of the stage 2 translation.

## Executing the ATS12NSOPW instruction

This instruction is executed 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>, c7, c8, 5	000	101	0111	1111	1000

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If EL3 is implemented and is using AArch64, execution of ATS12NSOPW in 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<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T7](#)==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 [HSTR\\_EL2.T7](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T7](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ATS12NSOUR, Address Translate Stages 1 and 2 Non-secure Only Unprivileged Read

The ATS12NSOUR characteristics are:

## Purpose

Performs stage 1 and 2 address translations as defined for PL0 and the Non-secure state, with permissions as if reading from the given virtual address.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

ATS12NSOUR is a 32-bit System instruction.

## Field descriptions

The ATS12NSOUR input value 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
Input address for translation																															

### Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This instruction takes a VA as input. The resulting address is the PA that is the output address of the stage 2 translation.

## Executing the ATS12NSOUR instruction

This instruction is executed 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>, c7, c8, 6	000	110	0111	1111	1000

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If EL3 is implemented and is using AArch64, execution of ATS12NSOUR in 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<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T7](#)==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 [HSTR\\_EL2.T7](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T7](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ATS12NSOUW, Address Translate Stages 1 and 2 Non-secure Only Unprivileged Write

The ATS12NSOUW characteristics are:

## Purpose

Performs stage 1 and 2 address translations as defined for PL0 and the Non-secure state, with permissions as if writing to the given virtual address.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

ATS12NSOUW is a 32-bit System instruction.

## Field descriptions

The ATS12NSOUW input value 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
Input address for translation																															

### Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This instruction takes a VA as input. The resulting address is the PA that is the output address of the stage 2 translation.

## Executing the ATS12NSOUW instruction

This instruction is executed 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>, c7, c8, 7	000	111	0111	1111	1000

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If EL3 is implemented and is using AArch64, execution of ATS12NSOUW in 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<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T7](#)==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 [HSTR\\_EL2.T7](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T7](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ATS1CPR, Address Translate Stage 1 Current state PL1 Read

The ATS1CPR characteristics are:

## Purpose

Performs stage 1 address translation as defined for PL1 and the current Security state, with permissions as if reading from the given virtual address.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

ATS1CPR is a 32-bit System instruction.

## Field descriptions

The ATS1CPR input value 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
Input address for translation																															

### Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This instruction takes a VA as input. In an implementation that includes EL2, when executed in Non-secure state, the resulting address is the IPA that is the output address of the stage 1 translation. Otherwise, the resulting address is a PA.

## Executing the ATS1CPR instruction

This instruction is executed 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>, c7, c8, 0	000	000	0111	1111	1000

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T7](#)=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 [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ATS1CPRP, Address Translate Stage 1 Current state PL1 Read PAN

The ATS1CPRP characteristics are:

## Purpose

When ARMv8.2-ATS1E1 is implemented, performs a stage 1 address translation at PL1 and in the current Security state, where the value of PSTATE.PAN determines if a read from a location will generate a permission fault for a privileged access.

This System instruction is part of the Address translation instructions functional group.

## Configuration

This instruction is introduced in ARMv8.2.

## Attributes

ATS1CPRP is a 32-bit System instruction.

## Field descriptions

The ATS1CPRP input value 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
Input address for translation																															

### Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This instruction takes a VA as input. In an implementation that includes EL2, when executed in Non-secure state, the resulting address is the IPA that is the output address of the stage 1 translation. Otherwise, the resulting address is a PA.

## Executing the ATS1CPRP instruction

This instruction is executed 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>, c7, c9, 0	000	000	0111	1111	1001

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T7==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 [HSTR\\_EL2.T7==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T7==1](#), Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ATS1CPW, Address Translate Stage 1 Current state PL1 Write

The ATS1CPW characteristics are:

## Purpose

Performs stage 1 address translation as defined for PL1 and the current Security state, with permissions as if writing to the given virtual address.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

ATS1CPW is a 32-bit System instruction.

## Field descriptions

The ATS1CPW input value 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
Input address for translation																															

### Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This instruction takes a VA as input. In an implementation that includes EL2, when executed in Non-secure state, the resulting address is the IPA that is the output address of the stage 1 translation. Otherwise, the resulting address is a PA.

## Executing the ATS1CPW instruction

This instruction is executed 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>, c7, c8, 1	000	001	0111	1111	1000

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T7](#)=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 [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ATS1CPWP, Address Translate Stage 1 Current state PL1 Write PAN

The ATS1CPWP characteristics are:

## Purpose

When ARMv8.2-ATS1E1 is implemented, performs a stage 1 address translation at PL1 and in the current Security state, where the value of PSTATE.PAN determines if a write to the location will generate a permission fault for a privileged access.

This System instruction is part of the Address translation instructions functional group.

## Configuration

This instruction is introduced in ARMv8.2.

## Attributes

ATS1CPWP is a 32-bit System instruction.

## Field descriptions

The ATS1CPWP input value 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
Input address for translation																															

### Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This instruction takes a VA as input. In an implementation that includes EL2, when executed in Non-secure state, the resulting address is the IPA that is the output address of the stage 1 translation. Otherwise, the resulting address is a PA.

## Executing the ATS1CPWP instruction

This instruction is executed 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>, c7, c9, 1	000	001	0111	1111	1001

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T7](#)=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 [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ATS1CUR, Address Translate Stage 1 Current state Unprivileged Read

The ATS1CUR characteristics are:

## Purpose

Performs stage 1 address translation as defined for PL0 and the current Security state, with permissions as if reading from the given virtual address.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

ATS1CUR is a 32-bit System instruction.

## Field descriptions

The ATS1CUR input value 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
Input address for translation																															

### Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This instruction takes a VA as input. In an implementation that includes EL2, when executed in Non-secure state, the resulting address is the IPA that is the output address of the stage 1 translation. Otherwise, the resulting address is a PA.

## Executing the ATS1CUR instruction

This instruction is executed 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>, c7, c8, 2	000	010	0111	1111	1000

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T7](#)=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 [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ATS1CUW, Address Translate Stage 1 Current state Unprivileged Write

The ATS1CUW characteristics are:

## Purpose

Performs stage 1 address translation as defined for PL0 and the current Security state, with permissions as if writing to the given virtual address.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

ATS1CUW is a 32-bit System instruction.

## Field descriptions

The ATS1CUW input value 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
Input address for translation																															

### Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This instruction takes a VA as input. In an implementation that includes EL2, when executed in Non-secure state, the resulting address is the IPA that is the output address of the stage 1 translation. Otherwise, the resulting address is a PA.

## Executing the ATS1CUW instruction

This instruction is executed 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>, c7, c8, 3	000	011	0111	1111	1000

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T7](#)=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 [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ATS1HR, Address Translate Stage 1 Hyp mode Read

The ATS1HR characteristics are:

## Purpose

Performs stage 1 address translation as defined for PL2 and the Non-secure state, with permissions as if reading from the given virtual address.

This System instruction is part of:

- The Address translation instructions functional group.
- The Virtualization registers functional group.

## Configuration

There are no configuration notes.

## Attributes

ATS1HR is a 32-bit System instruction.

## Field descriptions

The ATS1HR input value 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
Input address for translation																															

### Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This instruction takes a VA as input. The resulting address is the PA that is the output address of the translation.

## Executing the ATS1HR instruction

This instruction is executed 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>, c7, c8, 0	100	000	0111	1111	1000

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T7](#)=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 [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ATS1HW, Address Translate Stage 1 Hyp mode Write

The ATS1HW characteristics are:

## Purpose

Performs stage 1 address translation as defined for PL2 and the Non-secure state, with permissions as if writing to the given virtual address.

This System instruction is part of:

- The Address translation instructions functional group.
- The Virtualization registers functional group.

## Configuration

There are no configuration notes.

## Attributes

ATS1HW is a 32-bit System instruction.

## Field descriptions

The ATS1HW input value 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
Input address for translation																															

### Bits [31:0]

Input address for translation. The resulting address can be read from the [PAR](#).

This instruction takes a VA as input. The resulting address is the PA that is the output address of the translation.

## Executing the ATS1HW instruction

This instruction is executed 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>, c7, c8, 1	100	001	0111	1111	1000

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T7](#)=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 [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# BPIALL, Branch Predictor Invalidate All

The BPIALL characteristics are:

## Purpose

Invalidate all entries from branch predictors.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

In an implementation where the branch predictors are architecturally invisible, this instruction can execute as a NOP.

## Attributes

BPIALL is a 32-bit System instruction.

## Field descriptions

BPIALL ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the BPIALL instruction

This instruction is executed 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>, c7, c5, 6	000	110	0111	1111	0101

The PE ignores the value of <Rt>. Software does not have to write a value to this register before issuing this instruction.

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR.FB](#) is 1, at Non-secure EL1 this instruction executes as a [BPIALLIS](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T7](#)=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 [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# BPIALLIS, Branch Predictor Invalidate All, Inner Shareable

The BPIALLIS characteristics are:

## Purpose

Invalidate all entries from branch predictors Inner Shareable.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

In an implementation where the branch predictors are architecturally invisible, this instruction can execute as a NOP.

## Attributes

BPIALLIS is a 32-bit System instruction.

## Field descriptions

BPIALLIS ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the BPIALLIS instruction

This instruction is executed 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>, c7, c1, 6	000	110	0111	1111	0001

The PE ignores the value of <Rt>. Software does not have to write a value to this register before issuing this instruction.

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T7==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 [HSTR\\_EL2.T7==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T7==1](#), Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# BPIMVA, Branch Predictor Invalidate by VA

The BPIMVA characteristics are:

## Purpose

Invalidate virtual address from branch predictors.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

In an implementation where the branch predictors are architecturally invisible, this instruction can execute as a NOP.

## Attributes

BPIMVA is a 32-bit System instruction.

## Field descriptions

The BPIMVA input value 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
Virtual address to use																															

### Bits [31:0]

Virtual address to use.

## Executing the BPIMVA instruction

This instruction is executed 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>, c7, c5, 7	000	111	0111	1111	0101

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T7](#)=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 [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# CP15DMB, Data Memory Barrier System instruction

The CP15DMB characteristics are:

## Purpose

Performs a Data Memory Barrier.

ARM deprecates any use of this operation, and strongly recommends that software use the DMB instruction instead.

This System instruction is part of the Legacy feature registers functional group.

## Configuration

There are no configuration notes.

## Attributes

CP15DMB is a 32-bit System instruction.

## Field descriptions

CP15DMB ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the CP15DMB instruction

This instruction is executed 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>, c7, c10, 5	000	101	0111	1111	1010

## Accessibility

The instruction is executable 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 syntax that can be used to execute this instruction.

## 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 executing this System instruction.

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

- If [SCTLR](#).CP15BEN==0, execution of this instruction at PL0 and PL1 is UNDEFINED.
- If [SCTLR\\_EL1](#).CP15BEN==0, execution of this instruction at PL0 is UNDEFINED.
- If [HSCTLR](#).CP15BEN==0, execution of this instruction at PL2 is UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2](#).T7==1, Non-secure execution of this instruction at EL0 and 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 [HSTR\\_EL2](#).T7==1, Non-secure execution of this instruction at EL0 and EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR](#).T7==1, Non-secure execution of this instruction at EL0 and EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# CP15DSB, Data Synchronization Barrier System instruction

The CP15DSB characteristics are:

## Purpose

Performs a Data Synchronization Barrier.

ARM deprecates any use of this operation, and strongly recommends that software use the DSB instruction instead.

This System instruction is part of the Legacy feature registers functional group.

## Configuration

There are no configuration notes.

## Attributes

CP15DSB is a 32-bit System instruction.

## Field descriptions

CP15DSB ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the CP15DSB instruction

This instruction is executed 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>, c7, c10, 4	000	100	0111	1111	1010

## Accessibility

The instruction is executable 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 syntax that can be used to execute this instruction.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.



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

- If [SCTLR](#).CP15BEN==0, execution of this instruction at PL0 and PL1 is UNDEFINED.
- If [SCTLR\\_EL1](#).CP15BEN==0, execution of this instruction at PL0 is UNDEFINED.
- If [HSCTLR](#).CP15BEN==0, execution of this instruction at PL2 is UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2](#).T7==1, Non-secure execution of this instruction at EL0 and 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 [HSTR\\_EL2](#).T7==1, Non-secure execution of this instruction at EL0 and EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR](#).T7==1, Non-secure execution of this instruction at EL0 and EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# CP15ISB, Instruction Synchronization Barrier System instruction

The CP15ISB characteristics are:

## Purpose

Performs an Instruction Synchronization Barrier.

ARM deprecates any use of this operation, and strongly recommends that software use the ISB instruction instead.

This System instruction is part of the Legacy feature registers functional group.

## Configuration

There are no configuration notes.

## Attributes

CP15ISB is a 32-bit System instruction.

## Field descriptions

CP15ISB ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the CP15ISB instruction

This instruction is executed 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>, c7, c5, 4	000	100	0111	1111	0101

## Accessibility

The instruction is executable 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 syntax that can be used to execute this instruction.

## 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 executing this System instruction.

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

- If [SCTLR](#).CP15BEN==0, execution of this instruction at PL0 and PL1 is UNDEFINED.
- If [SCTLR\\_EL1](#).CP15BEN==0, execution of this instruction at PL0 is UNDEFINED.
- If [HSCTLR](#).CP15BEN==0, execution of this instruction at PL2 is UNDEFINED.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2](#).T7==1, Non-secure execution of this instruction at EL0 and 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 [HSTR\\_EL2](#).T7==1, Non-secure execution of this instruction at EL0 and EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR](#).T7==1, Non-secure execution of this instruction at EL0 and EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# DCCIMVAC, Data Cache line Clean and Invalidate by VA to PoC

The DCCIMVAC characteristics are:

## Purpose

Clean and Invalidate data or unified cache line by virtual address to PoC.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch32 System instruction DCCIMVAC performs the same function as AArch64 System instruction [DC CIVAC](#).

## Attributes

DCCIMVAC is a 32-bit System instruction.

## Field descriptions

The DCCIMVAC input value 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
Virtual address to use																															

### Bits [31:0]

Virtual address to use.

## Executing the DCCIMVAC instruction

This instruction is executed 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>, c7, c14, 1	000	001	0111	1111	1110

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 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 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.TPC](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2.T7](#)=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.TPC](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR.TPC](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# DCCISW, Data Cache line Clean and Invalidate by Set/Way

The DCCISW characteristics are:

## Purpose

Clean and Invalidate data or unified cache line by set/way.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch32 System instruction DCCISW performs the same function as AArch64 System instruction [DC C1SW](#).

## Attributes

DCCISW is a 32-bit System instruction.

## Field descriptions

The DCCISW input value 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
SetWay																												Level	0		

### SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$ ,  $L = \text{Log}_2(\text{LINELEN})$ ,  $B = (L + S)$ ,  $S = \text{Log}_2(\text{NSETS})$ .

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

### Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

### Bit [0]

Reserved, RES0.

## Executing the DCCISW instruction

This instruction is executed 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>, c7, c14, 2	000	010	0111	1111	1110

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is CONSTRAINED UNPREDICTABLE and one of the following occurs:

- The instruction is UNDEFINED
- The instruction performs cache maintenance on one of:
  - No cache lines.
  - A single arbitrary cache line.
  - Multiple arbitrary cache lines.

## 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HCR\\_EL2](#).TSW==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2](#).T7==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](#).TSW==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2](#).T7==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR](#).TSW==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR](#).T7==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# DCCMVAC, Data Cache line Clean by VA to PoC

The DCCMVAC characteristics are:

## Purpose

Clean data or unified cache line by virtual address to PoC.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch32 System instruction DCCMVAC performs the same function as AArch64 System instruction [DC CVAC](#).

## Attributes

DCCMVAC is a 32-bit System instruction.

## Field descriptions

The DCCMVAC input value 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
Virtual address to use																															

### Bits [31:0]

Virtual address to use.

## Executing the DCCMVAC instruction

This instruction is executed 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>, c7, c10, 1	000	001	0111	1111	1010

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 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 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.TPC](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2.T7](#)=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.TPC](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR.TPC](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# DCCMVAU, Data Cache line Clean by VA to PoU

The DCCMVAU characteristics are:

## Purpose

Clean data or unified cache line by virtual address to PoU.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch32 System instruction DCCMVAU performs the same function as AArch64 System instruction [DC CVAU](#).

## Attributes

DCCMVAU is a 32-bit System instruction.

## Field descriptions

The DCCMVAU input value 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
Virtual address to use																															

### Bits [31:0]

Virtual address to use.

## Executing the DCCMVAU instruction

This instruction is executed 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>, c7, c11, 1	000	001	0111	1111	1011

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 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 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](#).TPU==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2](#).T7==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](#).TPU==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2](#).T7==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR](#).TPU==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR](#).T7==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# DCCSW, Data Cache line Clean by Set/Way

The DCCSW characteristics are:

## Purpose

Clean data or unified cache line by set/way.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch32 System instruction DCCSW performs the same function as AArch64 System instruction [DC CSW](#).

## Attributes

DCCSW is a 32-bit System instruction.

## Field descriptions

The DCCSW input value 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
SetWay																												Level	0		

### SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$ ,  $L = \text{Log}_2(\text{LINELEN})$ ,  $B = (L + S)$ ,  $S = \text{Log}_2(\text{NSETS})$ .

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

### Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

### Bit [0]

Reserved, RES0.

## Executing the DCCSW instruction

This instruction is executed 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>, c7, c10, 2	000	010	0111	1111	1010

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is CONSTRAINED UNPREDICTABLE and one of the following occurs:

- The instruction is UNDEFINED
- The instruction performs cache maintenance on one of:
  - No cache lines.
  - A single arbitrary cache line.
  - Multiple arbitrary cache lines.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HCR\\_EL2](#).TSW==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2](#).T7==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](#).TSW==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2](#).T7==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR](#).TSW==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR](#).T7==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# DCIMVAC, Data Cache line Invalidate by VA to PoC

The DCIMVAC characteristics are:

## Purpose

Invalidate data or unified cache line by virtual address to PoC.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch32 System instruction DCIMVAC performs the same function as AArch64 System instruction [DCIVAC](#).

## Attributes

DCIMVAC is a 32-bit System instruction.

## Field descriptions

The DCIMVAC input value 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
Virtual address to use																															

### Bits [31:0]

Virtual address to use.

## Executing the DCIMVAC instruction

This instruction is executed 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>, c7, c6, 1	000	001	0111	1111	0110

It is IMPLEMENTATION DEFINED whether, when this instruction is executed, it can generate a watchpoint. If this instruction can generate a watchpoint this is prioritized in the same way as other watchpoints.

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

At EL1, this instruction performs a cache clean and invalidate, meaning it performs the same invalidation as a [DCCIMVAC](#) instruction, if all of the following apply:

- EL2 is implemented and either:
  - EL2 is using AArch64 and the value of [HCR\\_EL2.VM](#) is 1.
  - EL2 is using AArch32 and the value of [HCR.VM](#) is 1.
- Execution is in Non-secure state, or EL3 is not implemented.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 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.TPC](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2.T7](#)=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.TPC](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR.TPC](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# DCISW, Data Cache line Invalidate by Set/Way

The DCISW characteristics are:

## Purpose

Invalidate data or unified cache line by set/way.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch32 System instruction DCISW performs the same function as AArch64 System instruction [DC ISW](#).

## Attributes

DCISW is a 32-bit System instruction.

## Field descriptions

The DCISW input value 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
SetWay																												Level	0		

### SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$ ,  $L = \text{Log}_2(\text{LINELEN})$ ,  $B = (L + S)$ ,  $S = \text{Log}_2(\text{NSETS})$ .

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

### Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

### Bit [0]

Reserved, RES0.

## Executing the DCISW instruction

This instruction is executed 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>, c7, c6, 2	000	010	0111	1111	0110



## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

At EL1, this instruction performs a cache clean and invalidate, meaning it performs the same invalidation as a [DCCISW](#) instruction, if all of the following apply:

- EL2 is implemented and either:
  - EL2 is using AArch64 and the value of [HCR\\_EL2](#).{SWIO, VM} is not {0, 0}.
  - EL2 is using AArch32 and the value of [HCR](#).{SWIO, VM} is not {0, 0}.
- Execution is in Non-secure state, or EL3 is not implemented.

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is CONSTRAINED UNPREDICTABLE and one of the following occurs:

- The instruction is UNDEFINED
- The instruction performs cache maintenance on one of:
  - No cache lines.
  - A single arbitrary cache line.
  - Multiple arbitrary cache lines.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HCR\\_EL2](#).TSW==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2](#).T7==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](#).TSW==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2](#).T7==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR](#).TSW==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR](#).T7==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

# DTLBIALL, Data TLB Invalidate All

The DTLBIALL characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from data TLBs that are from any level of the translation table walk. The entries that are invalidated are as follows:

- If executed at Secure EL1 when EL3 is using AArch64, all entries that would be required for the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, all entries that would be required for the Secure PL1&0 translation regime.
- If executed at Non-secure EL1, all stage 1 translation table entries that would be required for the Non-secure PL1&0 translation regime and, if EL2 is implemented, they must match the current VMID.
- If executed at EL2, the stage 1 or stage 2 translation table entries that would be required for the Non-secure PL1&0 translation regime and matches the current VMID.

The invalidation only applies to the PE that executes this instruction.

ARM deprecates the use of this instruction. It is only provided for backwards compatibility with earlier versions of the ARM architecture.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

DTLBIALL is a 32-bit System instruction.

## Field descriptions

DTLBIALL ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the DTLBIALL instruction

This instruction is executed 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>, c8, c6, 0	000	000	1000	1111	0110

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBIALLIS](#) operating on data TLBs only.

## 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 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.
- If [HSTR\\_EL2.T8](#)=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.
- If [HSTR\\_EL2.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR.TTLB](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# DTLBIASID, Data TLB Invalidate by ASID match

The DTLBIASID characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from data TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
  - Is from a level of lookup above the final level.
  - Is a non-global entry from the final level of lookup.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this instruction.

ARM deprecates the use of this instruction. It is only provided for backwards compatibility with earlier versions of the ARM architecture.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

DTLBIASID is a 32-bit System instruction.

## Field descriptions

The DTLBIASID input value 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	ASID							

### Bits [31:8]

Reserved, RES0.

### ASID, bits [7:0]

ASID value to match. Any TLB entries for non-global pages that match the ASID values will be affected by this operation.

## Executing the DTLBIASID instruction

This instruction is executed 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>, c8, c6, 2	000	010	1000	1111	0110
-------------------------	-----	-----	------	------	------

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR](#).FB is 1, at Non-secure EL1 this instruction executes as a [TLBIASIDIS](#) operating on data TLBs only.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 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.
- If [HSTR\\_EL2](#).T8==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.
- If [HSTR\\_EL2](#).T8==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR](#).TTLB==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR](#).T8==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

# DTLBIMVA, Data TLB Invalidate by VA

The DTLBIMVA characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from data TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified address, and one of the following applies:
  - The entry is from a level of lookup above the final level and matches the specified ASID.
  - The entry is a global entry from the final level of lookup.
  - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this instruction.

ARM deprecates the use of this instruction. It is only provided for backwards compatibility with earlier versions of the ARM architecture.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

DTLBIMVA is a 32-bit System instruction.

## Field descriptions

The DTLBIMVA input value 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	0	0	ASID							

### VA, bits [31:12]

Virtual address to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

### Bits [11:8]

Reserved, RES0.

### ASID, bits [7:0]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

Global TLB entries that match the VA value will be affected by this operation, regardless of the value of the ASID field.

## Executing the DTLBIMVA instruction

This instruction is executed 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>, c8, c6, 1	000	001	1000	1111	0110

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBIMVAIS](#) operating on data TLBs only.

## 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 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.
- If [HSTR\\_EL2.T8](#)==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.
- If [HSTR\\_EL2.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

# ICIALLU, Instruction Cache Invalidate All to PoU

The ICIALLU characteristics are:

## Purpose

Invalidate all instruction caches to PoU. If branch predictors are architecturally visible, also flush branch predictors.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch32 System instruction ICIALLU performs the same function as AArch64 System instruction [IC IALLU](#).

## Attributes

ICIALLU is a 32-bit System instruction.

## Field descriptions

ICIALLU ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the ICIALLU instruction

This instruction is executed 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>, c7, c5, 0	000	000	0111	1111	0101

The PE ignores the value of <Rt>. Software does not have to write a value to this register before issuing this instruction.

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR.FB](#) is 1, at Non-secure EL1 this instruction executes as a [ICIALLUIS](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HCR\\_EL2](#).TPU==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2](#).T7==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](#).TPU==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2](#).T7==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR](#).TPU==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR](#).T7==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ICIALLUIS, Instruction Cache Invalidate All to PoU, Inner Shareable

The ICIALLUIS characteristics are:

## Purpose

Invalidate all instruction caches Inner Shareable to PoU. If branch predictors are architecturally visible, also flush branch predictors.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch32 System instruction ICIALLUIS performs the same function as AArch64 System instruction [IC IALLUIS](#).

## Attributes

ICIALLUIS is a 32-bit System instruction.

## Field descriptions

ICIALLUIS ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the ICIALLUIS instruction

This instruction is executed 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>, c7, c1, 0	000	000	0111	1111	0001

The PE ignores the value of <Rt>. Software does not have to write a value to this register before issuing this instruction.

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 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](#).TPU==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2](#).T7==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](#).TPU==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2](#).T7==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR](#).TPU==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR](#).T7==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ICIMVAU, Instruction Cache line Invalidate by VA to PoU

The ICIMVAU characteristics are:

## Purpose

Invalidate instruction cache line by virtual address to PoU.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch32 System instruction ICIMVAU performs the same function as AArch64 System instruction [IC IVAU](#).

## Attributes

ICIMVAU is a 32-bit System instruction.

## Field descriptions

The ICIMVAU input value 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
Virtual address to use																															

### Bits [31:0]

Virtual address to use.

## Executing the ICIMVAU instruction

This instruction is executed 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>, c7, c5, 1	000	001	0111	1111	0101

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 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 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.TPU](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2.T7](#)=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.TPU](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HSTR\\_EL2.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR.TPU](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T7](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ITLBIALL, Instruction TLB Invalidate All

The ITLBIALL characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from instruction TLBs that are from any level of the translation table walk. The entries that are invalidated are as follows:

- If executed at Secure EL1 when EL3 is using AArch64, all entries that would be required for the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, all entries that would be required for the Secure PL1&0 translation regime.
- If executed at Non-secure EL1, all stage 1 translation table entries that would be required for the Non-secure PL1&0 translation regime and, if EL2 is implemented, they must match the current VMID.
- If executed at EL2, the stage 1 or stage 2 translation table entries that would be required for the Non-secure PL1&0 translation regime and matches the current VMID.

The invalidation only applies to the PE that executes this instruction.

ARM deprecates the use of this instruction. It is only provided for backwards compatibility with earlier versions of the ARM architecture.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

ITLBIALL is a 32-bit System instruction.

## Field descriptions

ITLBIALL ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the ITLBIALL instruction

This instruction is executed 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>, c8, c5, 0	000	000	1000	1111	0101

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBIALLIS](#) operating on instruction TLBs only.

## 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 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.
- If [HSTR\\_EL2.T8](#)=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.
- If [HSTR\\_EL2.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR.TTLB](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# ITLBIASID, Instruction TLB Invalidate by ASID match

The ITLBIASID characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from instruction TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
  - Is from a level of lookup above the final level.
  - Is a non-global entry from the final level of lookup.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this instruction.

ARM deprecates the use of this instruction. It is only provided for backwards compatibility with earlier versions of the ARM architecture.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

ITLBIASID is a 32-bit System instruction.

## Field descriptions

The ITLBIASID input value 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	ASID									

### Bits [31:8]

Reserved, RES0.

### ASID, bits [7:0]

ASID value to match. Any TLB entries for non-global pages that match the ASID values will be affected by this operation.

## Executing the ITLBIASID instruction

This instruction is executed 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>, c8, c5, 2	000	010	1000	1111	0101
-------------------------	-----	-----	------	------	------

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR](#).FB is 1, at Non-secure EL1 this instruction executes as a [TLBIASIDIS](#) operating on instruction TLBs only.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 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.
- If [HSTR\\_EL2](#).T8==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.
- If [HSTR\\_EL2](#).T8==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR](#).TTLB==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR](#).T8==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

# ITLBIMVA, Instruction TLB Invalidate by VA

The ITLBIMVA characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from instruction TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified address, and one of the following applies:
  - The entry is from a level of lookup above the final level and matches the specified ASID.
  - The entry is a global entry from the final level of lookup.
  - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this instruction.

ARM deprecates the use of this instruction. It is only provided for backwards compatibility with earlier versions of the ARM architecture.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

ITLBIMVA is a 32-bit System instruction.

## Field descriptions

The ITLBIMVA input value 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	0	0	ASID							

### VA, bits [31:12]

Virtual address to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

### Bits [11:8]

Reserved, RES0.

### ASID, bits [7:0]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

Global TLB entries that match the VA value will be affected by this operation, regardless of the value of the ASID field.

## Executing the ITLBIMVA instruction

This instruction is executed 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>, c8, c5, 1	000	001	1000	1111	0101

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBIMVAIS](#) operating on instruction TLBs only.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 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.
- If [HSTR\\_EL2.T8](#)==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.
- If [HSTR\\_EL2.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

# TLBIALL, TLB Invalidate All

The TLBIALL characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from TLBs that are from any level of the translation table walk. The entries that are invalidated are as follows:

- If executed at Secure EL1 when EL3 is using AArch64, all entries that would be required for the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, all entries that would be required for the Secure PL1&0 translation regime.
- If executed at Non-secure EL1, all stage 1 translation table entries that would be required for the Non-secure PL1&0 translation regime and, if EL2 is implemented, they must match the current VMID.
- If executed at EL2, the stage 1 or stage 2 translation table entries that would be required for the Non-secure PL1&0 translation regime and matches the current VMID.

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

TLBIALL is a 32-bit System instruction.

## Field descriptions

TLBIALL ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBIALL instruction

This instruction is executed 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>, c8, c7, 0	000	000	1000	1111	0111

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBIALLIS](#).

## 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 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.
- If [HSTR\\_EL2.T8](#)=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.
- If [HSTR\\_EL2.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR.TTLB](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIALLH, TLB Invalidate All, Hyp mode

The TLBIALLH characteristics are:

## Purpose

If EL2 is implemented, invalidate all cached copies of translation table entries from TLBs that are from any level of the translation table walk that would be required for the Non-secure EL2 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.
- The Virtualization registers functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBIALLH is a 32-bit System instruction.

## Field descriptions

TLBIALLH ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBIALLH instruction

This instruction is executed 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>, c8, c7, 0	100	000	1000	1111	0111

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T8==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 [HSTR\\_EL2.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIALLHIS, TLB Invalidate All, Hyp mode, Inner Shareable

The TLBIALLHIS characteristics are:

## Purpose

If EL2 is implemented, invalidate all cached copies of translation table entries from TLBs that are from any level of the translation table walk that would be required for the Non-secure EL2 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.
- The Virtualization registers functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBIALLHIS is a 32-bit System instruction.

## Field descriptions

TLBIALLHIS ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBIALLHIS instruction

This instruction is executed 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>, c8, c3, 0	100	000	1000	1111	0011

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:



- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T8](#)=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 [HSTR\\_EL2.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIALLIS, TLB Invalidate All, Inner Shareable

The TLBIALLIS characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from TLBs that are from any level of the translation table walk. The entries that are invalidated are as follows:

- If executed at Secure EL1 when EL3 is using AArch64, all entries that would be required for the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, all entries that would be required for the Secure PL1&0 translation regime.
- If executed at Non-secure EL1, all stage 1 translation table entries that would be required for the Non-secure PL1&0 translation regime and, if EL2 is implemented, they must match the current VMID.
- If executed at EL2, the stage 1 or stage 2 translation table entries that would be required for the Non-secure PL1&0 translation regime and matches the current VMID.

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

TLBIALLIS is a 32-bit System instruction.

## Field descriptions

TLBIALLIS ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBIALLIS instruction

This instruction is executed 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>, c8, c3, 0	000	000	1000	1111	0011

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 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 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.
- If [HSTR\\_EL2.T8](#)=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.
- If [HSTR\\_EL2.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR.TTLB](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIALLSNH, TLB Invalidate All, Non-Secure Non-Hyp

The TLBIALLSNH characteristics are:

## Purpose

If EL2 is implemented, invalidate all cached copies of translation table entries from TLBs that are from any level of the translation table walk that would be required for stage 1 or stage 2 of the Non-secure PL1&0 translation regime, regardless of the associated VMID.

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

TLBIALLSNH is a 32-bit System instruction.

## Field descriptions

TLBIALLSNH ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBIALLSNH instruction

This instruction is executed 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>, c8, c7, 4	100	100	1000	1111	0111

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T8](#)=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 [HSTR\\_EL2.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIALLSNHS, TLB Invalidate All, Non-Secure Non-Hyp, Inner Shareable

The TLBIALLSNHS characteristics are:

## Purpose

If EL2 is implemented, invalidate all cached copies of translation table entries from TLBs that are from any level of the translation table walk that would be required for stage 1 or stage 2 of the Non-secure PL1&0 translation regime, regardless of the associated VMID.

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

TLBIALLSNHS is a 32-bit System instruction.

## Field descriptions

TLBIALLSNHS ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBIALLSNHS instruction

This instruction is executed 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>, c8, c3, 4	100	100	1000	1111	0011

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.

- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HSTR\\_EL2.T8](#)=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 [HSTR\\_EL2.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HSTR.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIASID, TLB Invalidate by ASID match

The TLBIASID characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
  - Is from a level of lookup above the final level.
  - Is a non-global entry from the final level of lookup.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this instruction.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBIASID is a 32-bit System instruction.

## Field descriptions

The TLBIASID input value 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	ASID									

### Bits [31:8]

Reserved, RES0.

### ASID, bits [7:0]

ASID value to match. Any TLB entries for non-global pages that match the ASID values will be affected by this operation.

## Executing the TLBIASID instruction

This instruction is executed 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>, c8, c7, 2	000	010	1000	1111	0111



## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBIASIDIS](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 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.
- If [HSTR\\_EL2.T8](#)=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.
- If [HSTR\\_EL2.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR.TTLB](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

# TLBIASIDIS, TLB Invalidate by ASID match, Inner Shareable

The TLBIASIDIS characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used for the specified ASID, and either:
  - Is from a level of lookup above the final level.
  - Is a non-global entry from the final level of lookup.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation 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

TLBIASIDIS is a 32-bit System instruction.

## Field descriptions

The TLBIASIDIS input value 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	ASID							

ASID

### Bits [31:8]

Reserved, RES0.

### ASID, bits [7:0]

ASID value to match. Any TLB entries for non-global pages that match the ASID values will be affected by this operation.

## Executing the TLBIASIDIS instruction

This instruction is executed 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>, c8, c3, 2	000	010	1000	1111	0011

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 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 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.
- If [HSTR\\_EL2.T8](#)==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.
- If [HSTR\\_EL2.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIIPAS2, TLB Invalidate by Intermediate Physical Address, Stage 2

The TLBIIPAS2 characteristics are:

## Purpose

If EL2 is implemented, invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 2 only translation table entry, from any level of the translation table walk.
- SCR.NS is 1.
- The entry would be used for the specified IPA.
- The entry would be used with the current VMID.
- The entry would be required for the PL1&0 translation regime.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation only applies to the PE that executes this instruction.

This System instruction is part of:

- The TLB maintenance instructions functional group.
- The Virtualization registers functional group.

## Configuration

### Note

This System instruction is not implemented in architecture versions before ARMv8.

## Attributes

TLBIIPAS2 is a 32-bit System instruction.

## Field descriptions

The TLBIIPAS2 input value 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																												

IPA[39:12]

### Bits [31:28]

Reserved, RES0.

### IPA[39:12], bits [27:0]

Bits[39:12] of the intermediate physical address to match.

## Executing the TLBIIPAS2 instruction

This instruction is executed 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>, c8, c4, 1	100	001	1000	1111	0100

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.

If SCR.NS is 0, this instruction is a NOP.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T8==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 [HSTR\\_EL2.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIIPAS2IS, TLB Invalidate by Intermediate Physical Address, Stage 2, Inner Shareable

The TLBIIPAS2IS characteristics are:

## Purpose

If EL2 is implemented, invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 2 only translation table entry, from any level of the translation table walk.
- SCR.NS is 1.
- The entry would be used for the specified IPA.
- The entry would be used with the current VMID.
- The entry would be required for the PL1&0 translation regime.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this instructions.

This System instruction is part of:

- The TLB maintenance instructions functional group.
- The Virtualization registers functional group.

## Configuration

### Note

This System instruction is not implemented in architecture versions before ARMv8.

## Attributes

TLBIIPAS2IS is a 32-bit System instruction.

## Field descriptions

The TLBIIPAS2IS input value 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																												

IPA[39:12]

### Bits [31:28]

Reserved, RES0.

### IPA[39:12], bits [27:0]

Bits[39:12] of the intermediate physical address to match.

## Executing the TLBIIPAS2IS instruction

This instruction is executed 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>, c8, c0, 1	100	001	1000	1111	0000

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.

If SCR.NS is 0, this instruction is a NOP.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T8==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 [HSTR\\_EL2.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIIPAS2L, TLB Invalidate by Intermediate Physical Address, Stage 2, Last level

The TLBIIPAS2L characteristics are:

## Purpose

If EL2 is implemented, invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 2 only translation table entry, from the final level of the translation table walk.
- SCR.NS is 1.
- The entry would be used for the specified IPA.
- The entry would be used with the current VMID.
- The entry would be required for the PL1&0 translation regime.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation only applies to the PE that executes this instruction.

This System instruction is part of:

- The TLB maintenance instructions functional group.
- The Virtualization registers functional group.

## Configuration

### Note

This System instruction is not implemented in architecture versions before ARMv8.

## Attributes

TLBIIPAS2L is a 32-bit System instruction.

## Field descriptions

The TLBIIPAS2L input value 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																												

IPA[39:12]

### Bits [31:28]

Reserved, RES0.

### IPA[39:12], bits [27:0]

Bits[39:12] of the intermediate physical address to match.

## Executing the TLBIIPAS2L instruction

This instruction is executed 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>, c8, c4, 5	100	101	1000	1111	0100

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.

If SCR.NS is 0, this instruction is a NOP.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T8==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 [HSTR\\_EL2.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIIPAS2LIS, TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, Inner Shareable

The TLBIIPAS2LIS characteristics are:

## Purpose

If EL2 is implemented, invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 2 only translation table entry, from the final level of the translation table walk.
- SCR.NS is 1.
- The entry would be used for the specified IPA.
- The entry would be used with the current VMID.
- The entry would be required for the PL1&0 translation regime.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this instructions.

This System instruction is part of:

- The TLB maintenance instructions functional group.
- The Virtualization registers functional group.

## Configuration

### Note

This System instruction is not implemented in architecture versions before ARMv8.

## Attributes

TLBIIPAS2LIS is a 32-bit System instruction.

## Field descriptions

The TLBIIPAS2LIS input value 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																												

IPA[39:12]

### Bits [31:28]

Reserved, RES0.

### IPA[39:12], bits [27:0]

Bits[39:12] of the intermediate physical address to match.

## Executing the TLBIIPAS2LIS instruction

This instruction is executed 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>, c8, c0, 5	100	101	1000	1111	0000

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.

If SCR.NS is 0, this instruction is a NOP.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T8==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 [HSTR\\_EL2.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIMVA, TLB Invalidate by VA

The TLBIMVA characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified address, and one of the following applies:
  - The entry is from a level of lookup above the final level and matches the specified ASID.
  - The entry is a global entry from the final level of lookup.
  - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this instruction.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBIMVA is a 32-bit System instruction.

## Field descriptions

The TLBIMVA input value 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 0 0				ASID															

### VA, bits [31:12]

Virtual address to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

### Bits [11:8]

Reserved, RES0.

### ASID, bits [7:0]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

Global TLB entries that match the VA value will be affected by this operation, regardless of the value of the ASID field.

## Executing the TLBIMVA instruction

This instruction is executed using MCR with the following syntax:

MCR &lt;syntax&gt;

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

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c8, c7, 1	000	001	1000	1111	0111

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBIMVAIS](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 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.
- If [HSTR\\_EL2.T8](#)==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.
- If [HSTR\\_EL2.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

# TLBIMVAA, TLB Invalidate by VA, All ASID

The TLBIMVAA characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- The entry would be used to translate the specified address.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this instruction.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBIMVAA is a 32-bit System instruction.

## Field descriptions

The TLBIMVAA input value 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	0	0	0	0	0	0	0	0	0	0

### VA, bits [31:12]

Virtual address to match. Any unlocked TLB entries that match the VA will be affected by this operation, regardless of the ASID.

### Bits [11:0]

Reserved, RES0.

## Executing the TLBIMVAA instruction

This instruction is executed 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>, c8, c7, 3	000	011	1000	1111	0111

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBIMVAAIS](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 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.
- If [HSTR\\_EL2.T8](#)=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.
- If [HSTR\\_EL2.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR.TTLB](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIMVAAIS, TLB Invalidate by VA, All ASID, Inner Shareable

The TLBIMVAAIS characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- The entry would be used to translate the specified address.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation 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

TLBIMVAAIS is a 32-bit System instruction.

## Field descriptions

The TLBIMVAAIS input value 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	0	0	0	0	0	0	0	0	0	0

### VA, bits [31:12]

Virtual address to match. Any unlocked TLB entries that match the VA will be affected by this operation, regardless of the ASID.

### Bits [11:0]

Reserved, RES0.

## Executing the TLBIMVAAIS instruction

This instruction is executed 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>, c8, c3, 3	000	011	1000	1111	0011



## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 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 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.
- If [HSTR\\_EL2](#).T8==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.
- If [HSTR\\_EL2](#).T8==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR](#).TTLB==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR](#).T8==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIMVAAL, TLB Invalidate by VA, All ASID, Last level

The TLBIMVAAL characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry, from the final level of the translation table walk.
- The entry would be used to translate the specified address.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this instruction.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

### Note

This System instruction is not implemented in architecture versions before ARMv8.

## Attributes

TLBIMVAAL is a 32-bit System instruction.

## Field descriptions

The TLBIMVAAL input value 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	0	0	0	0	0	0	0	0	0	0

### VA, bits [31:12]

Virtual address to match. Any unlocked TLB entries that match the VA will be affected by this operation, regardless of the ASID.

### Bits [11:0]

Reserved, RES0.

## Executing the TLBIMVAAL instruction

This instruction is executed 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>, c8, c7, 7	000	111	1000	1111	0111
-------------------------	-----	-----	------	------	------

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBIMVAALIS](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 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.
- If [HSTR\\_EL2.T8](#)=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.
- If [HSTR\\_EL2.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and  $\text{SCR\_EL3.NS}=1$  :

- If [HCR.TTLB](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)=1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

# TLBIMVAALIS, TLB Invalidate by VA, All ASID, Last level, Inner Shareable

The TLBIMVAALIS characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry, from the final level of the translation table walk.
- The entry would be used to translate the specified address.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation 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

### Note

This System instruction is not implemented in architecture versions before ARMv8.

## Attributes

TLBIMVAALIS is a 32-bit System instruction.

## Field descriptions

The TLBIMVAALIS input value 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	0	0	0	0	0	0	0	0	0	0

### VA, bits [31:12]

Virtual address to match. Any unlocked TLB entries that match the VA will be affected by this operation, regardless of the ASID.

### Bits [11:0]

Reserved, RES0.

## Executing the TLBIMVAALIS instruction

This instruction is executed 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>, c8, c3, 7	000	111	1000	1111	0011

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 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.
- If [HSTR\\_EL2.T8](#)==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.
- If [HSTR\\_EL2.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

# TLBIMVAH, TLB Invalidate by VA, Hyp mode

The TLBIMVAH characteristics are:

## Purpose

If EL2 is implemented, invalidate all cached copies of translation table entries from TLBs that are from any level of the translation table walk that would be required for the Non-secure EL2 translation regime and used to translate the specified address.

The invalidation only applies to the PE that executes this instruction.

This System instruction is part of:

- The TLB maintenance instructions functional group.
- The Virtualization registers functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBIMVAH is a 32-bit System instruction.

## Field descriptions

The TLBIMVAH input value 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	0	0	0	0	0	0	0	0	0	0

### VA, bits [31:12]

Virtual address to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

### Bits [11:0]

Reserved, RES0.

## Executing the TLBIMVAH instruction

This instruction is executed 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>, c8, c7, 1	100	001	1000	1111	0111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3

x	x	0	-	-	n/a	WO
x	0	1	-	-	WO	WO
x	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T8==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 [HSTR\\_EL2.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

# TLBIMVAHIS, TLB Invalidate by VA, Hyp mode, Inner Shareable

The TLBIMVAHIS characteristics are:

## Purpose

If EL2 is implemented, invalidate all cached copies of translation table entries from TLBs that are from any level of the translation table walk that would be required for the Non-secure EL2 translation regime and used to translate the specified address.

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.
- The Virtualization registers functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBIMVAHIS is a 32-bit System instruction.

## Field descriptions

The TLBIMVAHIS input value 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	0	0	0	0	0	0	0	0	0	0

### VA, bits [31:12]

Virtual address to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

### Bits [11:0]

Reserved, RES0.

## Executing the TLBIMVAHIS instruction

This instruction is executed 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>, c8, c3, 1	100	001	1000	1111	0011

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3



x	x	0	-	-	n/a	WO
x	0	1	-	-	WO	WO
x	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T8==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 [HSTR\\_EL2.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIMVAIS, TLB Invalidate by VA, Inner Shareable

The TLBIMVAIS characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified address, and one of the following applies:
  - The entry is from a level of lookup above the final level and matches the specified ASID.
  - The entry is a global entry from the final level of lookup.
  - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation 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

TLBIMVAIS is a 32-bit System instruction.

## Field descriptions

The TLBIMVAIS input value 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 0 0				ASID															

### VA, bits [31:12]

Virtual address to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

### Bits [11:8]

Reserved, RES0.

### ASID, bits [7:0]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

Global TLB entries that match the VA value will be affected by this operation, regardless of the value of the ASID field.

## Executing the TLBIMVAIS instruction

This instruction is executed using MCR with the following syntax:

MCR &lt;syntax&gt;

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

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c8, c3, 1	000	001	1000	1111	0011

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 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.
- If [HSTR\\_EL2.T8](#)==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.
- If [HSTR\\_EL2.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIMVAL, TLB Invalidate by VA, Last level

The TLBIMVAL characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified address, and one of the following applies:
  - The entry is a global entry from the final level of lookup.
  - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation only applies to the PE that executes this instruction.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

This System instruction is not implemented in architecture versions before ARMv8.

## Attributes

TLBIMVAL is a 32-bit System instruction.

## Field descriptions

The TLBIMVAL input value 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	0	0	ASID							

### VA, bits [31:12]

Virtual address to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

### Bits [11:8]

Reserved, RES0.

### ASID, bits [7:0]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

Global TLB entries that match the VA value will be affected by this operation, regardless of the value of the ASID field.

## Executing the TLBIMVAL instruction

This instruction is executed using MCR with the following syntax:

MCR &lt;syntax&gt;

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

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c8, c7, 5	000	101	1000	1111	0111

## Accessibility

The instruction is executable as follows:

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

This table applies to all syntax that can be used to execute this instruction.

When [HCR.FB](#) is 1, at Non-secure EL1 this instruction executes as a [TLBIMVALIS](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 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.
- If [HSTR\\_EL2.T8](#)==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.
- If [HSTR\\_EL2.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

# TLBIMVALH, TLB Invalidate by VA, Last level, Hyp mode

The TLBIMVALH characteristics are:

## Purpose

If EL2 is implemented, invalidate all cached copies of translation table entries from TLBs that are from the final level of the translation table walk that would be required for the Non-secure EL2 translation regime and used to translate the specified address.

The invalidation only applies to the PE that executes this instruction.

This System instruction is part of:

- The TLB maintenance instructions functional group.
- The Virtualization registers functional group.

## Configuration

This System instruction is not implemented in architecture versions before ARMv8.

## Attributes

TLBIMVALH is a 32-bit System instruction.

## Field descriptions

The TLBIMVALH input value 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	0	0	0	0	0	0	0	0	0	0

### VA, bits [31:12]

Virtual address to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

### Bits [11:0]

Reserved, RES0.

## Executing the TLBIMVALH instruction

This instruction is executed 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>, c8, c7, 5	100	101	1000	1111	0111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3

x	x	0	-	-	n/a	WO
x	0	1	-	-	WO	WO
x	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T8==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 [HSTR\\_EL2.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# TLBIMVALHIS, TLB Invalidate by VA, Last level, Hyp mode, Inner Shareable

The TLBIMVALHIS characteristics are:

## Purpose

If EL2 is implemented, invalidate all cached copies of translation table entries from TLBs that are from the final level of the translation table walk that would be required for the Non-secure EL2 translation regime and used to translate the specified address.

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.
- The Virtualization registers functional group.

## Configuration

This System instruction is not implemented in architecture versions before ARMv8.

## Attributes

TLBIMVALHIS is a 32-bit System instruction.

## Field descriptions

The TLBIMVALHIS input value 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	0	0	0	0	0	0	0	0	0	0

### VA, bits [31:12]

Virtual address to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

### Bits [11:0]

Reserved, RES0.

## Executing the TLBIMVALHIS instruction

This instruction is executed 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>, c8, c3, 5	100	101	1000	1111	0011

## Accessibility

The instruction is executable as follows:



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

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.

If this instruction is executed in a Secure privileged mode other than Monitor mode, then the behavior is CONSTRAINED UNPREDICTABLE, and one of the following behaviors must occur:

- The instruction is UNDEFINED.
- The instruction is treated as a NOP.
- The instruction executes as if it had been executed in Monitor mode.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section G1.11.2 (Exception priority order) in the *ARM<sup>®</sup> 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 executing this System instruction.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HSTR\\_EL2.T8==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 [HSTR\\_EL2.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HSTR.T8==1](#), Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

# TLBIMVALIS, TLB Invalidate by VA, Last level, Inner Shareable

The TLBIMVALIS characteristics are:

## Purpose

Invalidate all cached copies of translation table entries from TLBs that meet the following requirements:

- The entry is a stage 1 translation table entry.
- The entry would be used to translate the specified address, and one of the following applies:
  - The entry is a global entry from the final level of lookup.
  - The entry is a non-global entry from the final level of lookup that matches the specified ASID.
- If EL2 is implemented, the entry would be used with the current VMID.

From the entries that match these requirement, the entries that are invalidated are required for the following translation regime:

- If executed at Secure EL1 when EL3 is using AArch64, the Secure EL1&0 translation regime.
- If executed in Secure state when EL3 is using AArch32, the Secure PL1&0 translation regime.
- If executed in Non-secure state, the Non-secure PL1&0 translation regime.

The invalidation 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

This System instruction is not implemented in architecture versions before ARMv8.

## Attributes

TLBIMVALIS is a 32-bit System instruction.

## Field descriptions

The TLBIMVALIS input value 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	0	0	ASID							

### VA, bits [31:12]

Virtual address to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

### Bits [11:8]

Reserved, RES0.

### ASID, bits [7:0]

ASID value to match. Any TLB entries that match the ASID value and VA value will be affected by this operation.

Global TLB entries that match the VA value will be affected by this operation, regardless of the value of the ASID field.

## Executing the TLBIMVALIS instruction

This instruction is executed using MCR with the following syntax:

MCR &lt;syntax&gt;

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

<syntax>	opc1	opc2	CRn	coproc	CRm
p15, 0, <Rt>, c8, c3, 5	000	101	1000	1111	0011

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 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 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.
- If [HSTR\\_EL2.T8](#)==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.
- If [HSTR\\_EL2.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

When EL2 is implemented and is using AArch32 and SCR\_EL3.NS==1 :

- If [HCR.TTLB](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.
- If [HSTR.T8](#)==1, Non-secure execution of this instruction at EL1 is trapped to Hyp mode.

02/05/2017 15:43

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

# 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)

[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\\_HPIR0\\_EL1](#): Interrupt Controller Highest Priority Pending Interrupt Register 0

[ICC\\_HPIR1\\_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\\_HPIR0\\_EL1](#): Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0

[ICV\\_HPIR1\\_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\\_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

[MDSCR\\_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

[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

[PMSELR\\_EL0](#): Performance Monitors Event Counter Selection Register

[PMSWINC\\_EL0](#): Performance Monitors Software Increment register

[PMUSERENR\\_EL0](#): Performance Monitors User Enable Register

[PMXEVCNTR\\_EL0](#): Performance Monitors Selected Event Count Register

[PMXEVTYPER\\_EL0](#): Performance Monitors Selected Event Type Register

[REVIDR\\_EL1](#): Revision ID Register

[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

# ACTLR\_EL1, Auxiliary Control Register (EL1)

The ACTLR\_EL1 characteristics are:

## Purpose

Provides IMPLEMENTATION DEFINED configuration and control options for execution at EL1 and EL0.

**Note**

ARM recommends the contents of this register have no effect on the PE when [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, and instead the configuration and control fields are provided by the [ACTLR\\_EL2](#) register. This avoids the need for software to manage the contents of these register when switching between a Guest OS and a Host OS.

This register is part of:

- The Other system control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch64 System register ACTLR\_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ACTLR](#).

AArch64 System register ACTLR\_EL1 bits [63:32] are architecturally mapped to AArch32 System register [ACTLR2](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

ACTLR\_EL1 is a 64-bit register.

## Field descriptions

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

**IMPLEMENTATION DEFINED, bits [63:0]**

IMPLEMENTATION DEFINED.

## Accessing the ACTLR\_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
-------------	-----	-----	-----	-----	-----

ACTLR_EL1	11	000	0001	0000	001
-----------	----	-----	------	------	-----

## 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	RW	RW
0	1	1	-	n/a	RW	RW
1	0	1	-	RW	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 SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HCR\\_EL2.TACR](#)==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.TACR](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

# ACTLR\_EL2, Auxiliary Control Register (EL2)

The ACTLR\_EL2 characteristics are:

## Purpose

Provides IMPLEMENTATION DEFINED configuration and control options for EL2.

**Note**

ARM recommends the contents of this register are updated to apply to EL0 when [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, gaining configuration and control fields from the [ACTLR\\_EL1](#). This avoids the need for software to manage the contents of these register when switching between a Guest OS and a Host OS.

This register is part of:

- The Virtualization registers functional group.
- The Other system control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch64 System register ACTLR\_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HACTLR](#).

AArch64 System register ACTLR\_EL2 bits [63:32] are architecturally mapped to AArch32 System register [HACTLR2](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

ACTLR\_EL2 is a 64-bit register.

## Field descriptions

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

**IMPLEMENTATION DEFINED, bits [63:0]**

IMPLEMENTATION DEFINED.

## Accessing the ACTLR\_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
ACTLR_EL2	11	100	0001	0000	001

## 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.

02/05/2017 15:43

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

# ACTLR\_EL3, Auxiliary Control Register (EL3)

The ACTLR\_EL3 characteristics are:

## Purpose

Provides IMPLEMENTATION DEFINED configuration and control options for EL3.

This register is part of:

- The Other system control registers functional group.
- The Security registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

ACTLR\_EL3 is a 64-bit register.

## Field descriptions

The ACTLR\_EL3 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
												IMPLEMENTATION DEFINED																			
												IMPLEMENTATION DEFINED																			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

## Accessing the ACTLR\_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
ACTLR_EL3	11	110	0001	0000	001

## Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3

# ACTLR\_EL3, Auxiliary Control Register (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.

02/05/2017 15:43

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



# AFSR0\_EL1, Auxiliary Fault Status Register 0 (EL1)

The AFSR0\_EL1 characteristics are:

## Purpose

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

This register is part of:

- The Exception and fault handling registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch64 System register AFSR0\_EL1 is architecturally mapped to AArch32 System register [ADFSR](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

AFSR0\_EL1 is a 32-bit register.

## Field descriptions

The AFSR0\_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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the AFSR0\_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
AFSR0_EL1	11	000	0101	0001	000
AFSR0_EL12	11	101	0101	0001	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3

AFSR0_EL1	x	x	0	-	RW	n/a	RW
AFSR0_EL1	0	0	1	-	RW	RW	RW
AFSR0_EL1	0	1	1	-	n/a	RW	RW
AFSR0_EL1	1	0	1	-	RW	<a href="#">AFSR0_EL2</a>	RW
AFSR0_EL1	1	1	1	-	n/a	<a href="#">AFSR0_EL2</a>	RW
AFSR0_EL12	x	x	0	-	-	n/a	-
AFSR0_EL12	0	0	1	-	-	-	-
AFSR0_EL12	0	1	1	-	n/a	-	-
AFSR0_EL12	1	0	1	-	-	RW	RW
AFSR0_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic AFSR0\_EL1 or AFSR0\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

## 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.

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.

# AFSR0\_EL2, Auxiliary Fault Status Register 0 (EL2)

The AFSR0\_EL2 characteristics are:

## Purpose

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

This register is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch64 System register AFSR0\_EL2 is architecturally mapped to AArch32 System register [HADFESR](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

AFSR0\_EL2 is a 32-bit register.

## Field descriptions

The AFSR0\_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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the AFSR0\_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
AFSR0_EL2	11	100	0101	0001	000
AFSR0_EL1	11	000	0101	0001	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
AFSR0_EL2	x	x	0	-	-	n/a	RW
AFSR0_EL2	0	0	1	-	-	RW	RW
AFSR0_EL2	0	1	1	-	n/a	RW	RW
AFSR0_EL2	1	0	1	-	-	RW	RW
AFSR0_EL2	1	1	1	-	n/a	RW	RW
AFSR0_EL1	x	x	0	-	<a href="#">AFSR0_EL1</a>	n/a	<a href="#">AFSR0_EL1</a>
AFSR0_EL1	0	0	1	-	<a href="#">AFSR0_EL1</a>	<a href="#">AFSR0_EL1</a>	<a href="#">AFSR0_EL1</a>
AFSR0_EL1	0	1	1	-	n/a	<a href="#">AFSR0_EL1</a>	<a href="#">AFSR0_EL1</a>
AFSR0_EL1	1	0	1	-	<a href="#">AFSR0_EL1</a>	RW	<a href="#">AFSR0_EL1</a>
AFSR0_EL1	1	1	1	-	n/a	RW	<a href="#">AFSR0_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic AFSR0\_EL2 or AFSR0\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

02/05/2017 15:43

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

# AFSR0\_EL3, Auxiliary Fault Status Register 0 (EL3)

The AFSR0\_EL3 characteristics are:

## Purpose

Provides additional IMPLEMENTATION DEFINED fault status information for exceptions taken to EL3.

This register is part of:

- The Exception and fault handling registers functional group.
- The Security registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

AFSR0\_EL3 is a 32-bit register.

## Field descriptions

The AFSR0\_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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the AFSR0\_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
AFSR0_EL3	11	110	0101	0001	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

## AFSR0\_EL3, Auxiliary Fault Status Register 0 (EL3)

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

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

02/05/2017 15:43

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

# AFSR1\_EL1, Auxiliary Fault Status Register 1 (EL1)

The AFSR1\_EL1 characteristics are:

## Purpose

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

This register is part of:

- The Exception and fault handling registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch64 System register AFSR1\_EL1 is architecturally mapped to AArch32 System register [AIFSR](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

AFSR1\_EL1 is a 32-bit register.

## Field descriptions

The AFSR1\_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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the AFSR1\_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
AFSR1_EL1	11	000	0101	0001	001
AFSR1_EL12	11	101	0101	0001	001

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3

AFSR1_EL1	x	x	0	-	RW	n/a	RW
AFSR1_EL1	0	0	1	-	RW	RW	RW
AFSR1_EL1	0	1	1	-	n/a	RW	RW
AFSR1_EL1	1	0	1	-	RW	<a href="#">AFSR1_EL2</a>	RW
AFSR1_EL1	1	1	1	-	n/a	<a href="#">AFSR1_EL2</a>	RW
AFSR1_EL12	x	x	0	-	-	n/a	-
AFSR1_EL12	0	0	1	-	-	-	-
AFSR1_EL12	0	1	1	-	n/a	-	-
AFSR1_EL12	1	0	1	-	-	RW	RW
AFSR1_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic AFSR1\_EL1 or AFSR1\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

## 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.

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.



# AFSR1\_EL2, Auxiliary Fault Status Register 1 (EL2)

The AFSR1\_EL2 characteristics are:

## Purpose

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

This register is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch64 System register AFSR1\_EL2 is architecturally mapped to AArch32 System register [HAIFSR](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

AFSR1\_EL2 is a 32-bit register.

## Field descriptions

The AFSR1\_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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the AFSR1\_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
AFSR1_EL2	11	100	0101	0001	001
AFSR1_EL1	11	000	0101	0001	001

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
AFSR1_EL2	x	x	0	-	-	n/a	RW
AFSR1_EL2	0	0	1	-	-	RW	RW
AFSR1_EL2	0	1	1	-	n/a	RW	RW
AFSR1_EL2	1	0	1	-	-	RW	RW
AFSR1_EL2	1	1	1	-	n/a	RW	RW
AFSR1_EL1	x	x	0	-	<a href="#">AFSR1_EL1</a>	n/a	<a href="#">AFSR1_EL1</a>
AFSR1_EL1	0	0	1	-	<a href="#">AFSR1_EL1</a>	<a href="#">AFSR1_EL1</a>	<a href="#">AFSR1_EL1</a>
AFSR1_EL1	0	1	1	-	n/a	<a href="#">AFSR1_EL1</a>	<a href="#">AFSR1_EL1</a>
AFSR1_EL1	1	0	1	-	<a href="#">AFSR1_EL1</a>	RW	<a href="#">AFSR1_EL1</a>
AFSR1_EL1	1	1	1	-	n/a	RW	<a href="#">AFSR1_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic AFSR1\_EL2 or AFSR1\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

02/05/2017 15:43

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

# AFSR1\_EL3, Auxiliary Fault Status Register 1 (EL3)

The AFSR1\_EL3 characteristics are:

## Purpose

Provides additional IMPLEMENTATION DEFINED fault status information for exceptions taken to EL3.

This register is part of:

- The Exception and fault handling registers functional group.
- The Security registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

AFSR1\_EL3 is a 32-bit register.

## Field descriptions

The AFSR1\_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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the AFSR1\_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
AFSR1_EL3	11	110	0101	0001	001

## 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

## AFSR1\_EL3, Auxiliary Fault Status Register 1 (EL3)

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

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

02/05/2017 15:43

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

# AIDR\_EL1, Auxiliary ID Register

The AIDR\_EL1 characteristics are:

## Purpose

Provides IMPLEMENTATION DEFINED identification information.

The value of this register must be interpreted in conjunction with the value of [MIDR\\_EL1](#).

This register is part of:

- The Identification registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch64 System register AIDR\_EL1 is architecturally mapped to AArch32 System register [AIDR](#).

## Attributes

AIDR\_EL1 is a 32-bit register.

## Field descriptions

The AIDR\_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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the AIDR\_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
AIDR_EL1	11	001	0000	0000	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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- If [HCR\\_EL2.TID1](#)=1, Non-secure read 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.TID1](#)=1, Non-secure read accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

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

# AMAIR\_EL1, Auxiliary Memory Attribute Indirection Register (EL1)

The AMAIR\_EL1 characteristics are:

## Purpose

Provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by [MAIR\\_EL1](#).

This register is part of:

- The Virtual memory control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch64 System register AMAIR\_EL1 bits [31:0] are architecturally mapped to AArch32 System register [AMAIRO](#).

AArch64 System register AMAIR\_EL1 bits [63:32] are architecturally mapped to AArch32 System register [AMAIR1](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

AMAIR\_EL1 is a 64-bit register.

## Field descriptions

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

AMAIR\_EL1 is permitted to be cached in a TLB.

### IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

## Accessing the AMAIR\_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
AMAIR_EL1	11	000	1010	0011	000
AMAIR_EL12	11	101	1010	0011	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
AMAIR_EL1	x	x	0	-	RW	n/a	RW
AMAIR_EL1	0	0	1	-	RW	RW	RW
AMAIR_EL1	0	1	1	-	n/a	RW	RW
AMAIR_EL1	1	0	1	-	RW	<a href="#">AMAIR_EL2</a>	RW
AMAIR_EL1	1	1	1	-	n/a	<a href="#">AMAIR_EL2</a>	RW
AMAIR_EL12	x	x	0	-	-	n/a	-
AMAIR_EL12	0	0	1	-	-	-	-
AMAIR_EL12	0	1	1	-	n/a	-	-
AMAIR_EL12	1	0	1	-	-	RW	RW
AMAIR_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic AMAIR\_EL1 or AMAIR\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

## 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.

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.



# AMAIR\_EL2, Auxiliary Memory Attribute Indirection Register (EL2)

The AMAIR\_EL2 characteristics are:

## Purpose

Provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by [MAIR\\_EL2](#).

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch64 System register AMAIR\_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HAMAIRO](#).

AArch64 System register AMAIR\_EL2 bits [63:32] are architecturally mapped to AArch32 System register [HAMAIR1](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

AMAIR\_EL2 is a 64-bit register.

## Field descriptions

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

AMAIR\_EL2 is permitted to be cached in a TLB.

### IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

## Accessing the AMAIR\_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
AMAIR_EL2	11	100	1010	0011	000

AMAIR_EL1	11	000	1010	0011	000
-----------	----	-----	------	------	-----

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
AMAIR_EL2	x	x	0	-	-	n/a	RW
AMAIR_EL2	0	0	1	-	-	RW	RW
AMAIR_EL2	0	1	1	-	n/a	RW	RW
AMAIR_EL2	1	0	1	-	-	RW	RW
AMAIR_EL2	1	1	1	-	n/a	RW	RW
AMAIR_EL1	x	x	0	-	<a href="#">AMAIR_EL1</a>	n/a	<a href="#">AMAIR_EL1</a>
AMAIR_EL1	0	0	1	-	<a href="#">AMAIR_EL1</a>	<a href="#">AMAIR_EL1</a>	<a href="#">AMAIR_EL1</a>
AMAIR_EL1	0	1	1	-	n/a	<a href="#">AMAIR_EL1</a>	<a href="#">AMAIR_EL1</a>
AMAIR_EL1	1	0	1	-	<a href="#">AMAIR_EL1</a>	RW	<a href="#">AMAIR_EL1</a>
AMAIR_EL1	1	1	1	-	n/a	RW	<a href="#">AMAIR_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic AMAIR\_EL2 or AMAIR\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

02/05/2017 15:43

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

# AMAIR\_EL3, Auxiliary Memory Attribute Indirection Register (EL3)

The AMAIR\_EL3 characteristics are:

## Purpose

Provides IMPLEMENTATION DEFINED memory attributes for the memory regions specified by [MAIR\\_EL3](#).

This register is part of:

- The Virtual memory control registers functional group.
- The Security registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

AMAIR\_EL3 is a 64-bit register.

## Field descriptions

The AMAIR\_EL3 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
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

AMAIR\_EL3 is permitted to be cached in a TLB.

### IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

## Accessing the AMAIR\_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
AMAIR_EL3	11	110	1010	0011	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.

02/05/2017 15:43

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

# CCSIDR\_EL1, Current Cache Size ID Register

The CCSIDR\_EL1 characteristics are:

## Purpose

Provides information about the architecture of the currently selected cache.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register CCSIDR\_EL1 is architecturally mapped to AArch32 System register [CCSIDR](#).

The implementation includes one CCSIDR\_EL1 for each cache that it can access. [CSSELR\\_EL1](#) selects which Cache Size ID Register is accessible.

## Attributes

CCSIDR\_EL1 is a 32-bit register.

## Field descriptions

The CCSIDR\_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
UNKNOWN				NumSets																Associativity										LineSize	

### UNKNOWN, bits [31:28]

Reserved, UNKNOWN.

### NumSets, bits [27:13]

(Number of sets in cache) - 1, therefore a value of 0 indicates 1 set in the cache. The number of sets does not have to be a power of 2.

### Associativity, bits [12:3]

(Associativity of cache) - 1, therefore a value of 0 indicates an associativity of 1. The associativity does not have to be a power of 2.

### LineSize, bits [2:0]

( $\text{Log}_2(\text{Number of bytes in cache line})$ ) - 4. For example:

For a line length of 16 bytes:  $\text{Log}_2(16) = 4$ , LineSize entry = 0. This is the minimum line length.

For a line length of 32 bytes:  $\text{Log}_2(32) = 5$ , LineSize entry = 1.

---

#### Note

The parameters NumSets, Associativity, and LineSize in these registers define the architecturally visible parameters that are required for the cache maintenance by Set/Way instructions. They are not guaranteed to represent the actual microarchitectural features of a design. You cannot make any inference about the actual sizes of caches based on these parameters.

---

## Accessing the CCSIDR\_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
CCSIDR_EL1	11	001	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.

If [CSSELR\\_EL1](#).Level is programmed to a cache level that is not implemented, then on a read of the CCSIDR\_EL1 the behavior is CONSTRAINED UNPREDICTABLE, and can be one of the following:

- The CCSIDR\_EL1 read is treated as NOP.
- The CCSIDR\_EL1 read is UNDEFINED.
- The CCSIDR\_EL1 read returns an UNKNOWN value.

## 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 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](#).TID2==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

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

# CLIDR\_EL1, Cache Level ID Register

The CLIDR\_EL1 characteristics are:

## Purpose

Identifies the type of cache, or caches, that are implemented at each level and can be managed using the architected cache maintenance instructions that operate by set/way, up to a maximum of seven levels. Also identifies the Level of Coherence (LoC) and Level of Unification (LoU) for the cache hierarchy.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register CLIDR\_EL1 is architecturally mapped to AArch32 System register [CLIDR](#).

## Attributes

CLIDR\_EL1 is a 64-bit register.

## Field descriptions

The CLIDR\_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	ICB
ICB			LoUU			LoC			LoUIS			Ctype7			Ctype6			Ctype5			Ctype4			Ctype3			Ctype2			Ctype1		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

### Bits [63:33]

Reserved, RES0.

### ICB, bits [32:30]

Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions.

The possible values are:

ICB	Meaning
000	Not disclosed by this mechanism.
001	L1 cache is the highest Inner Cacheable level.
010	L2 cache is the highest Inner Cacheable level.
011	L3 cache is the highest Inner Cacheable level.
100	L4 cache is the highest Inner Cacheable level.
101	L5 cache is the highest Inner Cacheable level.
110	L6 cache is the highest Inner Cacheable level.
111	L7 cache is the highest Inner Cacheable level.

### LoUU, bits [29:27]

Level of Unification Uniprocessor for the cache hierarchy.

### LoC, bits [26:24]

Level of Coherence for the cache hierarchy.

**LoUIS, bits [23:21]**

Level of Unification Inner Shareable for the cache hierarchy.

**Ctype<n>, bits [3(n-1)+2:3(n-1)], for n = 1 to 7**

Cache Type fields. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy. Possible values of each field are:

Ctype<n>	Meaning
000	No cache.
001	Instruction cache only.
010	Data cache only.
011	Separate instruction and data caches.
100	Unified cache.

All other values are reserved.

If software reads the Cache Type fields from Ctype1 upwards, once it has seen a value of 000, no caches that can be managed using the architected cache maintenance instructions that operate by set/way exist at further-out levels of the hierarchy. So, for example, if Ctype3 is the first Cache Type field with a value of 000, the values of Ctype4 to Ctype7 must be ignored.

**Accessing the CLIDR\_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
CLIDR_EL1	11	001	0000	0000	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.TID2==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.TID2==1](#), Non-secure read accesses to this register from EL1 are trapped to EL2.





# CNTFRQ\_EL0, Counter-timer Frequency register

The CNTFRQ\_EL0 characteristics are:

## Purpose

This register is provided so that software can discover the frequency of the system counter. It must be programmed with this value as part of system initialization. The value of the register is not interpreted by hardware.

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

## Configuration

AArch64 System register CNTFRQ\_EL0 is architecturally mapped to AArch32 System register [CNTFRQ](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTFRQ\_EL0 is a 32-bit register.

## Field descriptions

The CNTFRQ\_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
Clock frequency																															

### Bits [31:0]

Clock frequency. Indicates the system counter clock frequency, in Hz.

## Accessing the CNTFRQ\_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
CNTFRQ_EL0	11	011	1110	0000	000

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
EL1 is the highest implemented Exception level	x	x	x	RO	RW	n/a	n/a
EL2 is the highest implemented Exception level	x	0	1	RO	RO	RW	n/a
EL2 is the highest implemented Exception level	x	1	1	RO	n/a	RW	n/a

EL3 is the highest implemented Exception level	x	x	0	RO	RO	RO	RW
EL3 is the highest implemented Exception level	x	0	1	RO	RO	RO	RW
EL3 is the highest implemented Exception level	x	1	1	RO	n/a	RO	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 HCR\_EL2.E2H==0 :

- If [CNTKCTL\\_EL1](#).EL0PCTEN==0, and [CNTKCTL\\_EL1](#).EL0VCTEN==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==1 && HCR\_EL2.TGE==0 :

- If [CNTKCTL\\_EL1](#).EL0PCTEN==0, and [CNTKCTL\\_EL1](#).EL0VCTEN==0, Non-secure 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==1 && HCR\_EL2.TGE==1 :

- If [CNTHCTL\\_EL2](#).EL0PCTEN==0, and [CNTHCTL\\_EL2](#).EL0VCTEN==0, Non-secure read accesses to this register from EL0 are trapped to EL2.

02/05/2017 15:43

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

# CNTHCTL\_EL2, Counter-timer Hypervisor Control register

The CNTHCTL\_EL2 characteristics are:

## Purpose

Controls the generation of an event stream from the physical counter, and access from Non-secure EL1 to the physical counter and the Non-secure EL1 physical timer.

This register is part of:

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

## Configuration

AArch64 System register CNTHCTL\_EL2 is architecturally mapped to AArch32 System register [CNTHCTL](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTHCTL\_EL2 is a 32-bit register.

## Field descriptions

The CNTHCTL\_EL2 bit assignments are:

### When HCR\_EL2.E2H == 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	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EVNTI		EVNTDIR		EVNTEN		EL1PCEN		EL1PCTEN			

This format applies in all ARMv8.0 implementations, and it also contains a description of the behavior when EL3 is implemented and EL2 is not implemented.

### Bits [31:8]

Reserved, RES0.

### EVNTI, bits [7:4]

Selects which bit (0 to 15) of the counter register [CNTPCT\\_EL0](#) is the trigger for the event stream generated from that counter, when that stream is enabled.

### EVNTDIR, bit [3]

Controls which transition of the counter register [CNTPCT\\_EL0](#) trigger bit, defined by EVNTI, generates an event when the event stream is enabled:

EVNTDIR	Meaning
0	A 0 to 1 transition of the trigger bit triggers an event.
1	A 1 to 0 transition of the trigger bit triggers an event.

### EVNTEN, bit [2]

Enables the generation of an event stream from the counter register CNTPCT\_EL0:

EVNTEN	Meaning
0	Disables the event stream.
1	Enables the event stream.

### EL1PCEN, bit [1]

Traps Non-secure EL0 and EL1 accesses to the physical timer registers to EL2.

EL1PCEN	Meaning
0	From AArch64 state: Non-secure EL0 and EL1 accesses to the <a href="#">CNTP_CTL_EL0</a> , <a href="#">CNTP_CVAL_EL0</a> , and <a href="#">CNTP_TVAL_EL0</a> are trapped to EL2, unless it is trapped by <a href="#">CNTKCTL_EL1.EL0PTEN</a> . From AArch32 state: Non-secure EL0 and EL1 accesses to the <a href="#">CNTP_CTL</a> , <a href="#">CNTP_CVAL</a> , and <a href="#">CNTP_TVAL</a> are trapped to EL2, unless it is trapped by <a href="#">CNTKCTL_EL1.EL0PTEN</a> or <a href="#">CNTKCTL.PL0PTEN</a> .
1	This control does not cause any instructions to be trapped.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 1 other than for the purpose of a direct read.

### EL1PCTEN, bit [0]

Traps Non-secure EL0 and EL1 accesses to the physical counter register to EL2.

EL1PCTEN	Meaning
0	From AArch64 state: Non-secure EL0 and EL1 accesses to the <a href="#">CNTPCT_EL0</a> are trapped to EL2, unless it is trapped by <a href="#">CNTKCTL_EL1</a> .EL0PCTEN. From AArch32 state: Non-secure EL0 and EL1 accesses to the <a href="#">CNTPCT</a> are trapped to EL2, unless it is trapped by <a href="#">CNTKCTL_EL1</a> .EL0PCTEN or <a href="#">CNTKCTL</a> .PL0PCTEN.
1	This control does not cause any instructions to be trapped.

If EL3 is implemented and EL2 is not implemented, behavior is as if this bit is 1 other than for the purpose of a direct read.

### When HCR\_EL2.E2H == 1:

3130292827262524232221201918171615141312	11	10	9	8	7	6	5	4	3	2	1	0
00000000000000000000000000	EL1PTEN	EL1PCTEN	EL0PTEN	EL0VTEN	EVNTI	EVNTDIR	EVNTEN	EL0VCTEN	EL0PC			

**Bits [31:12]**

Reserved, RES0.

### EL1PTEN, bit [11]

When [HCR\\_EL2.TGE](#) is 0, traps Non-secure EL0 and EL1 accesses to the physical timer registers to EL2.

EL1PTEN	Meaning
0	From AArch64 state: Non-secure EL0 and EL1 accesses to the <a href="#">CNTP_CTL_EL0</a> , <a href="#">CNTP_CVAL_EL0</a> , and <a href="#">CNTP_TVAL_EL0</a> are trapped to EL2, unless it is trapped by <a href="#">CNTKCTL_EL1</a> . EL0PTEN. From AArch32 state: Non-secure EL0 and EL1 accesses to the <a href="#">CNTP_CTL</a> , <a href="#">CNTP_CVAL</a> , and <a href="#">CNTP_TVAL</a> are trapped to EL2, unless it is trapped by <a href="#">CNTKCTL_EL1</a> . EL0PTEN or <a href="#">CNTKCTL</a> . PL0PTEN.
1	This control does not cause any instructions to be trapped.

When [HCR\\_EL2.TGE](#) is 1, this control does not cause any instructions to be trapped.

**EL1PCTEN, bit [10]**

When [HCR\\_EL2.TGE](#) is 0, traps Non-secure EL0 and EL1 accesses to the physical counter register to EL2.

EL1PCTEN	Meaning
0	From AArch64 state: Non-secure EL0 and EL1 accesses to the <a href="#">CNTPCT_EL0</a> are trapped to EL2, unless it is trapped by <a href="#">CNTKCTL_EL1.EL0PCTEN</a> . From AArch32 state: Non-secure EL0 and EL1 accesses to the <a href="#">CNTPCT</a> are trapped to EL2, unless it is trapped by <a href="#">CNTKCTL_EL1.EL0PCTEN</a> or <a href="#">CNTKCTL.PL0PCTEN</a> .
1	This control does not cause any instructions to be trapped.

When [HCR\\_EL2.TGE](#) is 1, this control does not cause any instructions to be trapped.

**EL0PTEN, bit [9]**

When [HCR\\_EL2.TGE](#) is 0, this control does not cause any instructions to be trapped.

When [HCR\\_EL2.TGE](#) is 1, traps EL0 accesses to the physical timer registers to EL2.

EL0PTEN	Meaning
0	EL0 using AArch64: EL0 accesses to the <a href="#">CNTP_CTL_EL0</a> , <a href="#">CNTP_CVAL_EL0</a> , and <a href="#">CNTP_TVAL_EL0</a> registers are trapped to EL2. EL0 using AArch32: EL0 accesses to the <a href="#">CNTP_CTL</a> , <a href="#">CNTP_CVAL</a> , and <a href="#">CNTP_TVAL</a> registers are trapped to EL2.
1	This control does not cause any instructions to be trapped.

**EL0VTEN, bit [8]**

When [HCR\\_EL2.TGE](#) is 0, this control does not cause any instructions to be trapped.

When [HCR\\_EL2.TGE](#) is 1, traps EL0 accesses to the virtual timer registers to EL2.

EL0VTEN	Meaning
0	EL0 using AArch64: EL0 accesses to the <a href="#">CNTV_CTL_EL0</a> , <a href="#">CNTV_CVAL_EL0</a> , and <a href="#">CNTV_TVAL_EL0</a> registers are trapped to EL2. EL0 using AArch32: EL0 accesses to the <a href="#">CNTV_CTL</a> , <a href="#">CNTV_CVAL</a> , and <a href="#">CNTV_TVAL</a> registers are trapped to EL2.
1	This control does not cause any instructions to be trapped.

**EVNTI, bits [7:4]**

Selects which bit (0 to 15) of the counter register [CNTPCT\\_EL0](#) is the trigger for the event stream generated from that counter, when that stream is enabled.

**EVNTDIR, bit [3]**

Controls which transition of the counter register [CNTPCT\\_EL0](#) trigger bit, defined by EVNTI, generates an event when the event stream is enabled:

EVNTDIR	Meaning
0	A 0 to 1 transition of the trigger bit triggers an event.
1	A 1 to 0 transition of the trigger bit triggers an event.

**EVNTEN, bit [2]**

Enables the generation of an event stream from the counter register [CNTPCT\\_EL0](#):

EVNTEN	Meaning
0	Disables the event stream.
1	Enables the event stream.

**EL0VCTEN, bit [1]**

When [HCR\\_EL2.TGE](#) is 0, this control does not cause any instructions to be trapped.

When [HCR\\_EL2.TGE](#) is 1, traps EL0 accesses to the frequency register and virtual counter register to EL2.

EL0VCTEN	Meaning
0	EL0 using AArch64: EL0 accesses to the <a href="#">CNTVCT_EL0</a> are trapped to EL2. EL0 using AArch64: EL0 accesses to the <a href="#">CNTFRQ_EL0</a> register are trapped to EL2, if <a href="#">CNTHCTL_EL2.EL0PCTEN</a> is also 0. EL0 using AArch32: EL0 accesses to the <a href="#">CNTVCT</a> are trapped to EL2. EL0 using AArch32: EL0 accesses to the <a href="#">CNTFRQ</a> register are trapped to EL2, if <a href="#">CNTHCTL_EL2.EL0PCTEN</a> is also 0.
1	This control does not cause any instructions to be trapped.

**EL0PCTEN, bit [0]**

When [HCR\\_EL2.TGE](#) is 0, this control does not cause any instructions to be trapped.

When [HCR\\_EL2.TGE](#) is 1, traps EL0 accesses to the frequency register and physical counter register to EL2.

EL0PCTEN	Meaning
0	EL0 using AArch64: EL0 accesses to the <a href="#">CNTPCT_EL0</a> are trapped to EL2. EL0 using AArch64: EL0 accesses to the <a href="#">CNTFRQ_EL0</a> register are trapped to EL2, if <a href="#">CNTHCTL_EL2.EL0VCTEN</a> is also 0. EL0 using AArch32: EL0 accesses to the <a href="#">CNTPCT</a> are trapped to EL2. EL0 using AArch32: EL0 accesses to the <a href="#">CNTFRQ</a> and register are trapped to EL2, if <a href="#">CNTHCTL_EL2.EL0VCTEN</a> is also 0.
1	This control does not cause any instructions to be trapped.

**Accessing the CNTHCTL\_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
CNTHCTL_EL2	11	100	1110	0001	000
CNTKCTL_EL1	11	000	1110	0001	000

**Accessibility**

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTHCTL_EL2	x	x	0	-	-	n/a	RW
CNTHCTL_EL2	0	0	1	-	-	RW	RW
CNTHCTL_EL2	0	1	1	-	n/a	RW	RW
CNTHCTL_EL2	1	0	1	-	-	RW	RW
CNTHCTL_EL2	1	1	1	-	n/a	RW	RW
CNTKCTL_EL1	x	x	0	-	<a href="#">CNTKCTL_EL1</a>	n/a	<a href="#">CNTKCTL_EL1</a>
CNTKCTL_EL1	0	0	1	-	<a href="#">CNTKCTL_EL1</a>	<a href="#">CNTKCTL_EL1</a>	<a href="#">CNTKCTL_EL1</a>
CNTKCTL_EL1	0	1	1	-	n/a	<a href="#">CNTKCTL_EL1</a>	<a href="#">CNTKCTL_EL1</a>

# CNTHCTL\_EL2, Counter-timer Hypervisor Control register

CNTKCTL_EL1	1	0	1	-	<a href="#">CNTKCTL_EL1</a>	RW	<a href="#">CNTKCTL_EL1</a>
CNTKCTL_EL1	1	1	1	-	n/a	RW	<a href="#">CNTKCTL_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic CNTHCTL\_EL2 or CNTKCTL\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

02/05/2017 15:43

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



# CNTHP\_CTL\_EL2, Counter-timer Hypervisor Physical Timer Control register

The CNTHP\_CTL\_EL2 characteristics are:

## Purpose

Control register 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\_CTL\_EL2 is architecturally mapped to AArch32 System register [CNTHP\\_CTL](#).

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

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTHP\_CTL\_EL2 is a 32-bit register.

## Field descriptions

The CNTHP\_CTL\_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
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	ISTATUS	IMASK	ENABLE

### Bits [31:3]

Reserved, RES0.

### ISTATUS, bit [2]

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0	Timer condition is not met.
1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.

For more information see 'Operation of the CompareValue views of the timers' and 'Operation of the TimerValue views of the timers' in the ARM ARM, chapter D6.

This bit is read-only.

### IMASK, bit [1]

Timer interrupt mask bit. Permitted values are:

IMASK	Meaning
0	Timer interrupt is not masked by the IMASK bit.
1	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

### ENABLE, bit [0]

Enables the timer. Permitted values are:

ENABLE	Meaning
0	Timer disabled.
1	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from [CNTHP\\_TVAL\\_EL2](#) continues to count down.

#### Note

Disabling the output signal might be a power-saving option.

## Accessing the CNTHP\_CTL\_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_CTL_EL2	11	100	1110	0010	001
CNTP_CTL_EL0	11	011	1110	0010	001

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTHP_CTL_EL2	x	x	0	-	-	n/a	RW
CNTHP_CTL_EL2	0	0	1	-	-	RW	RW
CNTHP_CTL_EL2	0	1	1	-	n/a	RW	RW
CNTHP_CTL_EL2	1	0	1	-	-	RW	RW
CNTHP_CTL_EL2	1	1	1	-	n/a	RW	RW
CNTP_CTL_EL0	x	x	0	<a href="#">CNTP_CTL_EL0</a>	<a href="#">CNTP_CTL_EL0</a>	n/a	<a href="#">CNTP_CTL_EL0</a>
CNTP_CTL_EL0	0	0	1	<a href="#">CNTP_CTL_EL0</a>	<a href="#">CNTP_CTL_EL0</a>	<a href="#">CNTP_CTL_EL0</a>	<a href="#">CNTP_CTL_EL0</a>
CNTP_CTL_EL0	0	1	1	<a href="#">CNTP_CTL_EL0</a>	n/a	<a href="#">CNTP_CTL_EL0</a>	<a href="#">CNTP_CTL_EL0</a>
CNTP_CTL_EL0	1	0	1	<a href="#">CNTP_CTL_EL0</a>	<a href="#">CNTP_CTL_EL0</a>	RW	RW
CNTP_CTL_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\_CTL\_EL2 or CNTP\_CTL\_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  $\text{SCR\_EL3.NS}=1 \ \&\& \ \text{HCR\_EL2.E2H}=1 \ \&\& \ \text{HCR\_EL2.TGE}=1$  :

- If [CNTHCTL\\_EL2](#).EL0PTEN==0, Non-secure accesses to this register from EL0 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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 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	<a href="#">CNTP_CVAL_EL0</a>	<a href="#">CNTP_CVAL_EL0</a>	n/a	<a href="#">CNTP_CVAL_EL0</a>
CNTP_CVAL_EL0	0	0	1	<a href="#">CNTP_CVAL_EL0</a>	<a href="#">CNTP_CVAL_EL0</a>	<a href="#">CNTP_CVAL_EL0</a>	<a href="#">CNTP_CVAL_EL0</a>
CNTP_CVAL_EL0	0	1	1	<a href="#">CNTP_CVAL_EL0</a>	n/a	<a href="#">CNTP_CVAL_EL0</a>	<a href="#">CNTP_CVAL_EL0</a>
CNTP_CVAL_EL0	1	0	1	<a href="#">CNTP_CVAL_EL0</a>	<a href="#">CNTP_CVAL_EL0</a>	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==1 && HCR\_EL2.TGE==1 :

- If [CNTHCTL\\_EL2.EL0PTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

# 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](#) - [CNTPTCT\\_EL0](#)).

On a write of this register, [CNTHP\\_CVAL\\_EL2](#) is set to ([CNTPTCT\\_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 ([CNTPTCT\\_EL0](#) - [CNTHP\\_CVAL\\_EL2](#)) is greater than 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 [CNTPTCT\\_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	<a href="#">CNTP_TVAL_EL0</a>	<a href="#">CNTP_TVAL_EL0</a>	n/a	<a href="#">CNTP_TVAL_EL0</a>
CNTP_TVAL_EL0	0	0	1	<a href="#">CNTP_TVAL_EL0</a>	<a href="#">CNTP_TVAL_EL0</a>	<a href="#">CNTP_TVAL_EL0</a>	<a href="#">CNTP_TVAL_EL0</a>
CNTP_TVAL_EL0	0	1	1	<a href="#">CNTP_TVAL_EL0</a>	n/a	<a href="#">CNTP_TVAL_EL0</a>	<a href="#">CNTP_TVAL_EL0</a>
CNTP_TVAL_EL0	1	0	1	<a href="#">CNTP_TVAL_EL0</a>	<a href="#">CNTP_TVAL_EL0</a>	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==1 && HCR\_EL2.TGE==1 :

- If [CNTHCTL\\_EL2.EL0PTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

# CNTHV\_CTL\_EL2, Counter-timer Virtual Timer Control register (EL2)

The CNTHV\_CTL\_EL2 characteristics are:

## Purpose

Control register for the EL2 virtual timer.

This register is part of the Generic Timer registers functional group.

## Configuration

AArch64 System register CNTHV\_CTL\_EL2 is architecturally mapped to AArch32 System register [CNTHV\\_CTL](#).

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\_CTL\_EL2 is a 32-bit register.

## Field descriptions

The CNTHV\_CTL\_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
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	ISTATUS	IMASK	ENABLE

### Bits [31:3]

Reserved, RES0.

### ISTATUS, bit [2]

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0	Timer condition is not met.
1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.

For more information see 'Operation of the CompareValue views of the timers' and 'Operation of the TimerValue views of the timers' in the ARM ARM, chapter D6.

This bit is read-only.

### IMASK, bit [1]

Timer interrupt mask bit. Permitted values are:



IMASK	Meaning
0	Timer interrupt is not masked by the IMASK bit.
1	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

### ENABLE, bit [0]

Enables the timer. Permitted values are:

ENABLE	Meaning
0	Timer disabled.
1	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from [CNTHV\\_TVAL\\_EL2](#) continues to count down.

#### Note

Disabling the output signal might be a power-saving option.

## Accessing the CNTHV\_CTL\_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_CTL_EL2	11	100	1110	0011	001
CNTV_CTL_EL0	11	011	1110	0011	001

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTHV_CTL_EL2	x	x	0	-	-	n/a	RW
CNTHV_CTL_EL2	0	0	1	-	-	RW	RW
CNTHV_CTL_EL2	0	1	1	-	n/a	RW	RW
CNTHV_CTL_EL2	1	0	1	-	-	RW	RW
CNTHV_CTL_EL2	1	1	1	-	n/a	RW	RW
CNTV_CTL_EL0	x	x	0	<a href="#">CNTV_CTL_EL0</a>	<a href="#">CNTV_CTL_EL0</a>	n/a	<a href="#">CNTV_CTL_EL0</a>
CNTV_CTL_EL0	0	0	1	<a href="#">CNTV_CTL_EL0</a>	<a href="#">CNTV_CTL_EL0</a>	<a href="#">CNTV_CTL_EL0</a>	<a href="#">CNTV_CTL_EL0</a>
CNTV_CTL_EL0	0	1	1	<a href="#">CNTV_CTL_EL0</a>	n/a	<a href="#">CNTV_CTL_EL0</a>	<a href="#">CNTV_CTL_EL0</a>
CNTV_CTL_EL0	1	0	1	<a href="#">CNTV_CTL_EL0</a>	<a href="#">CNTV_CTL_EL0</a>	RW	<a href="#">CNTV_CTL_EL0</a>
CNTV_CTL_EL0	1	1	1	RW	n/a	RW	<a href="#">CNTV_CTL_EL0</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic CNTHV\_CTL\_EL2 or CNTV\_CTL\_EL0 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

## 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==1 && HCR\_EL2.TGE==1 :

- If [CNTHCTL\\_EL2](#).EL0VTEN==0, Non-secure accesses to this register from EL0 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CompareValue																															

### CompareValue, bits [63:0]

Holds the EL2 virtual timer CompareValue.

When [CNTHV\\_CTL\\_EL2](#).ENABLE is 1, the timer condition is met when ([CNTVCT\\_EL0](#) - CompareValue) is greater than 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	<a href="#">CNTV_CVAL_EL0</a>	<a href="#">CNTV_CVAL_EL0</a>	n/a	<a href="#">CNTV_CVAL_EL0</a>
CNTV_CVAL_EL0	0	0	1	<a href="#">CNTV_CVAL_EL0</a>	<a href="#">CNTV_CVAL_EL0</a>	<a href="#">CNTV_CVAL_EL0</a>	<a href="#">CNTV_CVAL_EL0</a>
CNTV_CVAL_EL0	0	1	1	<a href="#">CNTV_CVAL_EL0</a>	n/a	<a href="#">CNTV_CVAL_EL0</a>	<a href="#">CNTV_CVAL_EL0</a>
CNTV_CVAL_EL0	1	0	1	<a href="#">CNTV_CVAL_EL0</a>	<a href="#">CNTV_CVAL_EL0</a>	RW	<a href="#">CNTV_CVAL_EL0</a>
CNTV_CVAL_EL0	1	1	1	RW	n/a	RW	<a href="#">CNTV_CVAL_EL0</a>

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==1 && HCR\_EL2.TGE==1 :

- If [CNTHCTL\\_EL2.EL0VTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

# 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 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 &lt;systemreg&gt;, &lt;Xt&gt;

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	<a href="#">CNTV_TVAL_EL0</a>	<a href="#">CNTV_TVAL_EL0</a>	n/a	<a href="#">CNTV_TVAL_EL0</a>
CNTV_TVAL_EL0	0	0	1	<a href="#">CNTV_TVAL_EL0</a>	<a href="#">CNTV_TVAL_EL0</a>	<a href="#">CNTV_TVAL_EL0</a>	<a href="#">CNTV_TVAL_EL0</a>
CNTV_TVAL_EL0	0	1	1	<a href="#">CNTV_TVAL_EL0</a>	n/a	<a href="#">CNTV_TVAL_EL0</a>	<a href="#">CNTV_TVAL_EL0</a>
CNTV_TVAL_EL0	1	0	1	<a href="#">CNTV_TVAL_EL0</a>	<a href="#">CNTV_TVAL_EL0</a>	RW	<a href="#">CNTV_TVAL_EL0</a>
CNTV_TVAL_EL0	1	1	1	RW	n/a	RW	<a href="#">CNTV_TVAL_EL0</a>

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==1 && HCR\_EL2.TGE==1 :

- If [CNTHCTL\\_EL2.EL0VTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

# CNTKCTL\_EL1, Counter-timer Kernel Control register

The CNTKCTL\_EL1 characteristics are:

## Purpose

When ARMv8.1-VHE is not implemented, or when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, this register controls the generation of an event stream from the virtual counter, and access from EL0 to the physical counter, virtual counter, EL1 physical timers, and the virtual timer.

When ARMv8.1-VHE is implemented and [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, this register does not cause any event stream from the virtual counter to be generated, and does not control access to the counters and timers. The access to counters and timers at EL0 is controlled by [CNTHCTL\\_EL2](#).

This register is part of the Generic Timer registers functional group.

## Configuration

AArch64 System register CNTKCTL\_EL1 is architecturally mapped to AArch32 System register [CNTKCTL](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTKCTL\_EL1 is a 32-bit register.

## Field descriptions

The CNTKCTL\_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	EL0PTEN	EL0VTEN	EVNTI	EVNTDIR	EVNTEN	EL0VCTEN	EL0PCTEN			

### Bits [31:10]

Reserved, RES0.

### EL0PTEN, bit [9]

When ARMv8.1-VHE is not implemented, or when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, traps EL0 accesses to the physical timer registers to EL1.

EL0PTEN	Meaning
0	EL0 using AArch64: EL0 accesses to the <a href="#">CNTP_CTL_EL0</a> , <a href="#">CNTP_CVAL_EL0</a> , and <a href="#">CNTP_TVAL_EL0</a> registers are trapped to EL1. EL0 using AArch32: EL0 accesses to the <a href="#">CNTP_CTL</a> , <a href="#">CNTP_CVAL</a> , and <a href="#">CNTP_TVAL</a> registers are trapped to EL1. When <a href="#">HCR_EL2</a> .TGE is 1, this trap is routed to EL2.
1	This control does not cause any instructions to be trapped.

When ARMv8.1-VHE is implemented and [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, this control does not cause any instructions to be trapped.

### EL0VTEN, bit [8]

When ARMv8.1-VHE is not implemented, or when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, traps EL0 accesses to the virtual timer registers to EL1.

EL0VTEN	Meaning
0	EL0 using AArch64: EL0 accesses to the <a href="#">CNTV_CTL_EL0</a> , <a href="#">CNTV_CVAL_EL0</a> , and <a href="#">CNTV_TVAL_EL0</a> registers are trapped to EL1. EL0 using AArch32: EL0 accesses to the <a href="#">CNTV_CTL</a> , <a href="#">CNTV_CVAL</a> , and <a href="#">CNTV_TVAL</a> registers are trapped to EL1. When <a href="#">HCR_EL2</a> .TGE is 1, this trap is routed to EL2.
1	This control does not cause any instructions to be trapped.

When ARMv8.1-VHE is implemented and [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, this control does not cause any instructions to be trapped.

#### EVNTI, bits [7:4]

Selects which bit (0 to 15) of the counter register [CNTVCT\\_EL0](#) is the trigger for the event stream generated from that counter, when that stream is enabled.

#### EVNTDIR, bit [3]

Controls which transition of the counter register [CNTVCT\\_EL0](#) trigger bit, defined by EVNTI, generates an event when the event stream is enabled:

EVNTDIR	Meaning
0	A 0 to 1 transition of the trigger bit triggers an event.
1	A 1 to 0 transition of the trigger bit triggers an event.

#### EVNTEN, bit [2]

When ARMv8.1-VHE is not implemented, or when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, enables the generation of an event stream from the counter register [CNTVCT\\_EL0](#):

EVNTEN	Meaning
0	Disables the event stream.
1	Enables the event stream.

When ARMv8.1-VHE is implemented and [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, this control does not enable the event stream.

#### EL0VCTEN, bit [1]

When ARMv8.1-VHE is not implemented, or when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, traps EL0 accesses to the frequency register and virtual counter register to EL1.

EL0VCTEN	Meaning
0	EL0 using AArch64: EL0 accesses to the <a href="#">CNTVCT_EL0</a> are trapped to EL1. EL0 using AArch64: EL0 accesses to the <a href="#">CNTFRQ_EL0</a> register are trapped to EL1, if <a href="#">CNTKCTL_EL1</a> .EL0PCTEN is also 0. EL0 using AArch32: EL0 accesses to the <a href="#">CNTVCT</a> are trapped to EL1. EL0 using AArch32: EL0 accesses to the <a href="#">CNTFRQ</a> register are trapped to EL1, if <a href="#">CNTKCTL_EL1</a> .EL0PCTEN is also 0. When <a href="#">HCR_EL2</a> .TGE is 1, this trap is routed to EL2.
1	This control does not cause any instructions to be trapped.

When ARMv8.1-VHE is implemented and [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, this control does not cause any instructions to be trapped.

#### EL0PCTEN, bit [0]

When ARMv8.1-VHE is not implemented, or when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, traps EL0 accesses to the frequency register and physical counter register to EL1.



EL0PCTEN	Meaning
0	<p>EL0 using AArch64: EL0 accesses to the <a href="#">CNTPCT_EL0</a> are trapped to EL1.</p> <p>EL0 using AArch64: EL0 accesses to the <a href="#">CNTFRQ_EL0</a> register are trapped to EL1, if <a href="#">CNTKCTL_EL1.EL0VCTEN</a> is also 0.</p> <p>EL0 using AArch32: EL0 accesses to the <a href="#">CNTPCT</a> are trapped to EL1.</p> <p>EL0 using AArch32: EL0 accesses to the <a href="#">CNTFRQ</a> and register are trapped to EL1, if <a href="#">CNTKCTL_EL1.EL0VCTEN</a> is also 0.</p> <p>When <a href="#">HCR_EL2.TGE</a> is 1, this trap is routed to EL2.</p>
1	This control does not cause any instructions to be trapped.

When ARMv8.1-VHE is implemented and [HCR\\_EL2.{E2H, TGE}](#) is {1, 1}, this control does not cause any instructions to be trapped.

## Accessing the CNTKCTL\_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
CNTKCTL_EL1	11	000	1110	0001	000
CNTKCTL_EL12	11	101	1110	0001	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTKCTL_EL1	x	x	0	-	RW	n/a	RW
CNTKCTL_EL1	0	0	1	-	RW	RW	RW
CNTKCTL_EL1	0	1	1	-	n/a	RW	RW
CNTKCTL_EL1	1	0	1	-	RW	<a href="#">CNTHCTL_EL2</a>	RW
CNTKCTL_EL1	1	1	1	-	n/a	<a href="#">CNTHCTL_EL2</a>	RW
CNTKCTL_EL12	x	x	0	-	-	n/a	-
CNTKCTL_EL12	0	0	1	-	-	-	-
CNTKCTL_EL12	0	1	1	-	n/a	-	-
CNTKCTL_EL12	1	0	1	-	-	RW	RW
CNTKCTL_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic CNTKCTL\_EL1 or CNTKCTL\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

# CNTPCT\_EL0, Counter-timer Physical Count register

The CNTPCT\_EL0 characteristics are:

## Purpose

Holds the 64-bit physical count value.

This register is part of the Generic Timer registers functional group.

## Configuration

AArch64 System register CNTPCT\_EL0 is architecturally mapped to AArch32 System register [CNTPCT](#).

## Attributes

CNTPCT\_EL0 is a 64-bit register.

## Field descriptions

The CNTPCT\_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
Physical count value																															
Physical count value																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Physical count value.

## Accessing the CNTPCT\_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
CNTPCT_EL0	11	011	1110	0000	001

## 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.

When  $\text{HCR\_EL2.E2H}==0$  :

- If [CNTKCTL\\_EL1](#).EL0PCTEN==0, read accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}==1$  &&  $\text{HCR\_EL2.E2H}==0$  :

- If [CNTHCTL\\_EL2](#).EL1PCTEN==0, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [CNTHCTL\\_EL2](#).EL1PCTEN==0, and [CNTKCTL\\_EL1](#).EL0PCTEN==1, Non-secure read accesses to this register from EL0 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 [CNTHCTL\\_EL2](#).EL1PCTEN==0, Non-secure read accesses to this register from EL1 are trapped to EL2.
- If [CNTHCTL\\_EL2](#).EL1PCTEN==0, and [CNTKCTL\\_EL1](#).EL0PCTEN==1, Non-secure read accesses to this register from EL0 are trapped to EL2.
- If [CNTKCTL\\_EL1](#).EL0PCTEN==0, Non-secure read accesses to this register from EL0 are trapped to EL1.

When EL2 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}==1$  &&  $\text{HCR\_EL2.E2H}==1$  &&  $\text{HCR\_EL2.TGE}==1$  :

- If [CNTHCTL\\_EL2](#).EL0PCTEN==0, Non-secure read accesses to this register from EL0 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTPS\_CTL\_EL1, Counter-timer Physical Secure Timer Control register

The CNTPS\_CTL\_EL1 characteristics are:

## Purpose

Control register for the secure physical timer, usually accessible at EL3 but configurably accessible at EL1 in Secure state.

This register is part of the Generic Timer registers functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTPS\_CTL\_EL1 is a 32-bit register.

## Field descriptions

The CNTPS\_CTL\_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	ISTATUS	IMASK	ENABLE

### Bits [31:3]

Reserved, RES0.

### ISTATUS, bit [2]

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0	Timer condition is not met.
1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.

For more information see 'Operation of the CompareValue views of the timers' and 'Operation of the TimerValue views of the timers' in the ARM ARM, chapter D6.

This bit is read-only.

### IMASK, bit [1]

Timer interrupt mask bit. Permitted values are:

IMASK	Meaning
0	Timer interrupt is not masked by the IMASK bit.
1	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

**ENABLE, bit [0]**

Enables the timer. Permitted values are:

ENABLE	Meaning
0	Timer disabled.
1	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from [CNTPS\\_TVAL\\_EL1](#) continues to count down.

**Note**

Disabling the output signal might be a power-saving option.

**Accessing the CNTPS\_CTL\_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_CTL_EL1	11	111	1110	0010	001

**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<sup>®</sup> 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.

# 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														
																	CompareValue														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### CompareValue, bits [63:0]

Holds the secure physical timer CompareValue.

When [CNTPS\\_CTL\\_EL1](#).ENABLE is 1, the timer condition is met when ([CNTPCT\\_EL0](#) - CompareValue) is greater than 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTP\_CTL\_EL0, Counter-timer Physical Timer Control register

The CNTP\_CTL\_EL0 characteristics are:

## Purpose

Control register for the EL1 physical timer.  
This register is part of the Generic Timer registers functional group.

## Configuration

AArch64 System register CNTP\_CTL\_EL0 is architecturally mapped to AArch32 System register [CNTP\\_CTL](#).  
RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTP\_CTL\_EL0 is a 32-bit register.

## Field descriptions

The CNTP\_CTL\_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	0	0	<a href="#">ISTATUS</a>	<a href="#">IMASK</a>	<a href="#">ENABLE</a>

### Bits [31:3]

Reserved, RES0.

### ISTATUS, bit [2]

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0	Timer condition is not met.
1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.

For more information see 'Operation of the CompareValue views of the timers' and 'Operation of the TimerValue views of the timers' in the ARM ARM, chapter D6.

This bit is read-only.

### IMASK, bit [1]

Timer interrupt mask bit. Permitted values are:

IMASK	Meaning
0	Timer interrupt is not masked by the IMASK bit.
1	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

**ENABLE, bit [0]**

Enables the timer. Permitted values are:

ENABLE	Meaning
0	Timer disabled.
1	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from [CNTP\\_TVAL\\_EL0](#) continues to count down.

**Note**

Disabling the output signal might be a power-saving option.

**Accessing the CNTP\_CTL\_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_CTL_EL0	11	011	1110	0010	001
CNTP_CTL_EL02	11	101	1110	0010	001

**Accessibility**

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTP_CTL_EL0	x	x	0	RW	RW	n/a	RW
CNTP_CTL_EL0	0	0	1	RW	RW	RW	RW
CNTP_CTL_EL0	0	1	1	RW	n/a	RW	RW
CNTP_CTL_EL0	1	0	1	RW	RW	<a href="#">CNTHP_CTL_EL2</a>	RW
CNTP_CTL_EL0	1	1	1	<a href="#">CNTHP_CTL_EL2</a>	n/a	<a href="#">CNTHP_CTL_EL2</a>	RW
CNTP_CTL_EL02	x	x	0	-	-	n/a	-
CNTP_CTL_EL02	0	0	1	-	-	-	-
CNTP_CTL_EL02	0	1	1	-	n/a	-	-
CNTP_CTL_EL02	1	0	1	-	-	RW	RW
CNTP_CTL_EL02	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic CNTP\_CTL\_EL0 or CNTP\_CTL\_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.

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.

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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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																															
31	30	29	28	27	26	25	24	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 EL1 physical timer CompareValue.

When [CNTP\\_CTL\\_EL0.ENABLE](#) is 1, the timer condition is met when ([CNTPCT\\_EL0](#) - CompareValue) is greater than 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	<a href="#">CNTHP_CVAL_EL2</a>	RW
CNTP_CVAL_EL0	1	1	1	<a href="#">CNTHP_CVAL_EL2</a>	n/a	<a href="#">CNTHP_CVAL_EL2</a>	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.

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.

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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTP\_TVAL\_EL0, Counter-timer Physical Timer TimerValue register

The CNTP\_TVAL\_EL0 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\_EL0 is architecturally mapped to AArch32 System register [CNTP\\_TVAL](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTP\_TVAL\_EL0 is a 32-bit register.

## Field descriptions

The CNTP\_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 physical timer.

On a read of this register:

- If [CNTP\\_CTL\\_EL0.ENABLE](#) is 0, the value returned is UNKNOWN.
- If [CNTP\\_CTL\\_EL0.ENABLE](#) is 1, the value returned is ([CNTP\\_CVAL\\_EL0](#) - [CNTPCT\\_EL0](#)).

On a write of this register, [CNTP\\_CVAL\\_EL0](#) is set to ([CNTPCT\\_EL0](#) + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When [CNTP\\_CTL\\_EL0.ENABLE](#) is 1, the timer condition is met when ([CNTPCT\\_EL0](#) - [CNTP\\_CVAL\\_EL0](#)) is greater than zero. This means that TimerValue acts like a 32-bit downcounter 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, so the TimerValue view appears to continue to count down.

## Accessing the CNTP\_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
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	<a href="#">CNTHP_TVAL_EL2</a>	RW
CNTP_TVAL_EL0	1	1	1	<a href="#">CNTHP_TVAL_EL2</a>	n/a	<a href="#">CNTHP_TVAL_EL2</a>	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.

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.

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.



# CNTVCT\_EL0, Counter-timer Virtual Count register

The CNTVCT\_EL0 characteristics are:

## Purpose

Holds the 64-bit virtual count value. The virtual count value is equal to the physical count value visible in [CNTPCT\\_EL0](#) minus the virtual offset visible in [CNTVOFF\\_EL2](#).

This register is part of the Generic Timer registers functional group.

## Configuration

AArch64 System register CNTVCT\_EL0 is architecturally mapped to AArch32 System register [CNTVCT](#).

The value of this register is the same as the value of [CNTPCT\\_EL0](#) in the following conditions:

- When EL2 is not implemented.
- When EL2 is implemented, [HCR\\_EL2.E2H](#) is 1, and this register is read from EL2.
- When EL2 is implemented, [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, and this register is read from Non-secure EL0 or EL2.

## Attributes

CNTVCT\_EL0 is a 64-bit register.

## Field descriptions

The CNTVCT\_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
Virtual count value																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Virtual count value																															

### Bits [63:0]

Virtual count value.

## Accessing the CNTVCT\_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
CNTVCT_EL0	11	011	1110	0000	010

## 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.

When HCR\_EL2.E2H==0 :

- If [CNTKCTL\\_EL1.EL0VCTEN](#)==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==1 && HCR\_EL2.TGE==0 :

- If [CNTKCTL\\_EL1.EL0VCTEN](#)==0, Non-secure 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==1 && HCR\_EL2.TGE==1 :

- If [CNTHCTL\\_EL2.EL0PCTEN](#)==0, Non-secure read accesses to this register from EL0 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTVOFF\_EL2, Counter-timer Virtual Offset register

The CNTVOFF\_EL2 characteristics are:

## Purpose

Holds the 64-bit virtual offset. This is the offset between the physical count value visible in [CNTPCT\\_ELO](#) and the virtual count value visible in [CNTVCT\\_ELO](#).

This register is part of:

- The Generic Timer registers functional group.
- The Virtualization registers functional group.

## Configuration

AArch64 System register CNTVOFF\_EL2 is architecturally mapped to AArch32 System register [CNTVOFF](#).

If EL2 is not implemented, this register is RES0 from EL3 and the virtual counter uses a fixed virtual offset of zero.

---

### Note

When EL2 is implemented and is using AArch64, the virtual counter uses a fixed virtual offset of zero in the following situations:

- [HCR\\_EL2.E2H](#) is 1, and [CNTVCT\\_ELO](#) is read from EL2.
  - [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, and either:
    - [CNTVCT\\_ELO](#) is read from Non-secure EL0 or EL2.
    - [CNTVCT](#) is read from Non-secure EL0.
- 

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTVOFF\_EL2 is a 64-bit register.

## Field descriptions

The CNTVOFF\_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
																	Virtual offset														
																	Virtual offset														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Virtual offset.

## Accessing the CNTVOFF\_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
CNTVOFF_EL2	11	100	1110	0000	011

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTV\_CTL\_EL0, Counter-timer Virtual Timer Control register

The CNTV\_CTL\_EL0 characteristics are:

## Purpose

Control register for the virtual timer.

This register is part of the Generic Timer registers functional group.

## Configuration

AArch64 System register CNTV\_CTL\_EL0 is architecturally mapped to AArch32 System register [CNTV\\_CTL](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CNTV\_CTL\_EL0 is a 32-bit register.

## Field descriptions

The CNTV\_CTL\_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	0	0	ISTATUS	IMASK	ENABLE

### Bits [31:3]

Reserved, RES0.

### ISTATUS, bit [2]

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0	Timer condition is not met.
1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.

For more information see 'Operation of the CompareValue views of the timers' and 'Operation of the TimerValue views of the timers' in the ARM ARM, chapter D6.

This bit is read-only.

### IMASK, bit [1]

Timer interrupt mask bit. Permitted values are:

IMASK	Meaning
0	Timer interrupt is not masked by the IMASK bit.
1	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

**ENABLE, bit [0]**

Enables the timer. Permitted values are:

ENABLE	Meaning
0	Timer disabled.
1	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from [CNTV\\_TVAL\\_EL0](#) continues to count down.

**Note**

Disabling the output signal might be a power-saving option.

**Accessing the CNTV\_CTL\_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_CTL_EL0	11	011	1110	0011	001
CNTV_CTL_EL02	11	101	1110	0011	001

**Accessibility**

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CNTV_CTL_EL0	x	x	0	RW	RW	n/a	RW
CNTV_CTL_EL0	0	0	1	RW	RW	RW	RW
CNTV_CTL_EL0	0	1	1	RW	n/a	RW	RW
CNTV_CTL_EL0	1	0	1	RW	RW	<a href="#">CNTHV_CTL_EL2</a>	RW
CNTV_CTL_EL0	1	1	1	<a href="#">CNTHV_CTL_EL2</a>	n/a	<a href="#">CNTHV_CTL_EL2</a>	RW
CNTV_CTL_EL02	x	x	0	-	-	n/a	-
CNTV_CTL_EL02	0	0	1	-	-	-	-
CNTV_CTL_EL02	0	1	1	-	n/a	-	-
CNTV_CTL_EL02	1	0	1	-	-	RW	RW
CNTV_CTL_EL02	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic CNTV\_CTL\_EL0 or CNTV\_CTL\_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==1 && HCR\_EL2.TGE==0 :

- If [CNTKCTL\\_EL1](#).EL0VTEN==0, Non-secure 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==1 && HCR\_EL2.TGE==1 :

- If [CNTHCTL\\_EL2](#).EL0VTEN==0, Non-secure accesses to this register from EL0 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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																															
31	30	29	28	27	26	25	24	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 EL1 virtual timer CompareValue.

When [CNTV\\_CTL\\_EL0.ENABLE](#) is 1, the timer condition is met when ([CNTVCT\\_EL0](#) - CompareValue) is greater than 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	<a href="#">CNTHV_CVAL_EL2</a>	RW
CNTV_CVAL_EL0	1	1	1	<a href="#">CNTHV_CVAL_EL2</a>	n/a	<a href="#">CNTHV_CVAL_EL2</a>	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<sup>®</sup> 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==1 && HCR\_EL2.TGE==0 :

- If [CNTKCTL\\_EL1.EL0VTEN](#)==0, Non-secure 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==1 && HCR\_EL2.TGE==1 :

- If [CNTHCTL\\_EL2.EL0VTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

# 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  $(\text{CNTV\_CVAL\_EL0} - (\text{CNTPCT\_EL0} - \text{CNTVOFF\_EL2}))$ .

On a write of this register, [CNTV\\_CVAL\\_EL0](#) is set to  $(\text{CNTPCT\_EL0} + \text{TimerValue})$ , where TimerValue is treated as a signed 32-bit integer.

When [CNTV\\_CTL\\_EL0.ENABLE](#) is 1, the timer condition is met when  $(\text{CNTPCT\_EL0} - \text{CNTV\_CVAL\_EL0})$  is greater than 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	<a href="#">CNTHV_TVAL_EL2</a>	RW
CNTV_TVAL_EL0	1	1	1	<a href="#">CNTHV_TVAL_EL2</a>	n/a	<a href="#">CNTHV_TVAL_EL2</a>	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==1 && HCR\_EL2.TGE==0 :

- If [CNTKCTL\\_EL1.EL0VTEN](#)==0, Non-secure 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==1 && HCR\_EL2.TGE==1 :

- If [CNTHCTL\\_EL2.EL0VTEN](#)==0, Non-secure accesses to this register from EL0 are trapped to EL2.

# CONTEXTIDR\_EL1, Context ID Register (EL1)

The CONTEXTIDR\_EL1 characteristics are:

## Purpose

Identifies the current Process Identifier.

The value of the whole of this register is called the Context ID and is used by:

- The debug logic, for Linked and Unlinked Context ID matching.
- The trace logic, to identify the current process.

The significance of this register is for debug and trace use only.

This register is used:

- In ARMv8.0.
- When ARMv8.1-VHE is implemented and [HCR\\_EL2.E2H](#) is 0.

---

### Note

When ARMv8.1-VHE is implemented and [HCR\\_EL2.E2H](#) is set to 1, [CONTEXTIDR\\_EL2](#) is used.

---

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch64 System register CONTEXTIDR\_EL1 is architecturally mapped to AArch32 System register [CONTEXTIDR](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CONTEXTIDR\_EL1 is a 32-bit register.

## Field descriptions

The CONTEXTIDR\_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
																PROCID															

### PROCID, bits [31:0]

Process Identifier. This field must be programmed with a unique value that identifies the current process.

---

### Note

In AArch32 state, when [TTBCR.EAE](#) is set to 0, [CONTEXTIDR](#).ASID holds the ASID.

In AArch64 state, CONTEXTIDR\_EL1 is independent of the ASID, and for the EL1&0 translation regime either [TTBR0\\_EL1](#) or [TTBR1\\_EL1](#) holds the ASID.

---

## Accessing the CONTEXTIDR\_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
CONTEXTIDR_EL1	11	000	1101	0000	001
CONTEXTIDR_EL12	11	101	1101	0000	001

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CONTEXTIDR_EL1	x	x	0	-	RW	n/a	RW
CONTEXTIDR_EL1	0	0	1	-	RW	RW	RW
CONTEXTIDR_EL1	0	1	1	-	n/a	RW	RW
CONTEXTIDR_EL1	1	0	1	-	RW	<a href="#">CONTEXTIDR_EL2</a>	RW
CONTEXTIDR_EL1	1	1	1	-	n/a	<a href="#">CONTEXTIDR_EL2</a>	RW
CONTEXTIDR_EL12	x	x	0	-	-	n/a	-
CONTEXTIDR_EL12	0	0	1	-	-	-	-
CONTEXTIDR_EL12	0	1	1	-	n/a	-	-
CONTEXTIDR_EL12	1	0	1	-	-	RW	RW
CONTEXTIDR_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic CONTEXTIDR\_EL1 or CONTEXTIDR\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

## 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.

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.

# CONTEXTIDR\_EL2, Context ID Register (EL2)

The CONTEXTIDR\_EL2 characteristics are:

## Purpose

When ARMv8.1-VHE is implemented and [HCR\\_EL2.E2H](#) is set to 1, identifies the current Process Identifier.

The value of the whole of this register is called the Context ID and is used by:

- The debug logic, for Linked and Unlinked Context ID matching.
- The trace logic, to identify the current process.

The significance of this register is for debug and trace use only.

---

### Note

In ARMv8.0, or when ARMv8.1-VHE is implemented and [HCR\\_EL2.E2H](#) is 0, [CONTEXTIDR\\_EL1](#) is used.

---

This register is part of the Virtual memory control registers functional group.

## Configuration

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

CONTEXTIDR\_EL2 is a 32-bit register.

## Field descriptions

The CONTEXTIDR\_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
																PROCID															

### PROCID, bits [31:0]

Process Identifier. This field must be programmed with a unique value that identifies the current process.

---

### Note

In AArch32 state, when [TTBCR.EAE](#) is set to 0, [CONTEXTIDR.ASID](#) holds the ASID.

In AArch64 state, CONTEXTIDR\_EL2 is independent of the ASID, and for the EL2&0 translation regime either [TTBR0\\_EL2](#) or [TTBR1\\_EL2](#) holds the ASID.

---

## Accessing the CONTEXTIDR\_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
CONTEXTIDR_EL2	11	100	1101	0000	001
CONTEXTIDR_EL1	11	000	1101	0000	001

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CONTEXTIDR_EL2	x	x	0	-	-	n/a	RW
CONTEXTIDR_EL2	0	0	1	-	-	RW	RW
CONTEXTIDR_EL2	0	1	1	-	n/a	RW	RW
CONTEXTIDR_EL2	1	0	1	-	-	RW	RW
CONTEXTIDR_EL2	1	1	1	-	n/a	RW	RW
CONTEXTIDR_EL1	x	x	0	-	<a href="#">CONTEXTIDR_EL1</a>	n/a	<a href="#">CONTEXTIDR_EL1</a>
CONTEXTIDR_EL1	0	0	1	-	<a href="#">CONTEXTIDR_EL1</a>	<a href="#">CONTEXTIDR_EL1</a>	<a href="#">CONTEXTIDR_EL1</a>
CONTEXTIDR_EL1	0	1	1	-	n/a	<a href="#">CONTEXTIDR_EL1</a>	<a href="#">CONTEXTIDR_EL1</a>
CONTEXTIDR_EL1	1	0	1	-	<a href="#">CONTEXTIDR_EL1</a>	RW	<a href="#">CONTEXTIDR_EL1</a>
CONTEXTIDR_EL1	1	1	1	-	n/a	RW	<a href="#">CONTEXTIDR_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic CONTEXTIDR\_EL2 or CONTEXTIDR\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

# CPACR\_EL1, Architectural Feature Access Control Register

The CPACR\_EL1 characteristics are:

## Purpose

Controls access to trace, SVE, Advanced SIMD and floating-point functionality.

This register is part of the Other system control registers functional group.

## Configuration

AArch64 System register CPACR\_EL1 is architecturally mapped to AArch32 System register [CPACR](#).

When [HCR\\_EL2](#).{E2H, TGE} == {1, 1}, the fields in this register have no effect on execution at EL0 and EL1. In this case, the controls provided by [CPTR\\_EL2](#) are used.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CPACR\_EL1 is a 32-bit register.

## Field descriptions

The CPACR\_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	<a href="#">TTA</a>	0	0	0	0	0	0	<a href="#">FPEN</a>	0	0	<a href="#">ZEN</a>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### Bits [31:29]

Reserved, RES0.

### TTA, bit [28]

Traps EL0 and EL1 System register accesses to all implemented trace registers to EL1, or to EL2 when [SCR\\_EL3](#).NS is 1 and [HCR\\_EL2](#).TGE is 1, from both Execution states.

TTA	Meaning
0	This control does not cause any instructions to be trapped.
1	This control causes EL0 and EL1 System register accesses to all implemented trace registers to be trapped.

### Note

- The ETMv4 architecture does not permit EL0 to access the trace registers. If the ARMv8-A architecture is implemented with an ETMv4 implementation, EL0 accesses to the trace registers are UNDEFINED, and any resulting exception is higher priority than an exception that would be generated because the value of [CPACR\\_EL1](#).TTA is 1.
- The ARMv8-A architecture does not provide traps on trace register accesses through the optional memory-mapped interface.

System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.

If System register access to the trace functionality is not implemented, this bit is RES0.



**Bits [27:22]**

Reserved, RES0.

**FPEN, bits [21:20]**

Traps EL0 and EL1 accesses to the SVE, Advanced SIMD, and floating-point registers to EL1, or to EL2 when [SCR\\_EL3.NS](#) is 1 and [HCR\\_EL2.TGE](#) is 1, from both Execution states, unless SVE is implemented and they are trapped by [CPACR\\_EL1.ZEN](#).

FPEN	Meaning
00	This control causes any instructions at EL0 or EL1 that use the registers associated with SVE, Advanced SIMD and floating-point execution to be trapped.
01	This control causes any instructions at EL0 that use the registers associated with SVE, Advanced SIMD and floating-point execution to be trapped, but does not cause any instruction in EL1 to be trapped.
10	This control causes any instructions at EL0 or EL1 that use the registers associated with SVE, Advanced SIMD and floating-point execution to be trapped.
11	This control does not cause any instructions to be trapped.

Writes to [MVFR0](#), [MVFR1](#) and [MVFR2](#) from EL1 or higher are CONSTRAINED UNPREDICTABLE and whether these accesses can be trapped by this control depends on implemented CONSTRAINED UNPREDICTABLE behavior.

**Note**

- Attempts to write to the FPSID count as use of the registers for accesses from EL1 or higher.
- Accesses from EL0 to [FPSID](#), [MVFR0](#), [MVFR1](#), [MVFR2](#), and [FPEXC](#) are UNDEFINED, and any resulting exception is higher priority than an exception that would be generated because the value of [CPACR\\_EL1.FPEN](#) is not 11.

**Bits [19:18]**

Reserved, RES0.

**ZEN, bits [17:16]**

In ARMv8.2:

Traps SVE instructions and instructions that access SVE System registers at EL0 and EL1 to EL1, or to EL2 when [SCR\\_EL3.NS](#) and [HCR\\_EL2.TGE](#) are both 1. Defined values are:

ZEN	Meaning
00	This control causes these instructions executed at EL0 or EL1 to be trapped.
01	This control causes these instructions executed at EL0 to be trapped, but does not cause any instruction in EL1 to be trapped.
10	This control causes these instructions executed at EL0 or EL1 to be trapped.
11	This control does not cause any instruction to be trapped.

If SVE is not implemented, this field is RES0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

**Bits [15:0]**

Reserved, RES0.

## Accessing the CPACR\_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
CPACR_EL1	11	000	0001	0000	010
CPACR_EL12	11	101	0001	0000	010

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CPACR_EL1	x	x	0	-	RW	n/a	RW
CPACR_EL1	0	0	1	-	RW	RW	RW
CPACR_EL1	0	1	1	-	n/a	RW	RW
CPACR_EL1	1	0	1	-	RW	<a href="#">CPTR_EL2</a>	RW
CPACR_EL1	1	1	1	-	n/a	<a href="#">CPTR_EL2</a>	RW
CPACR_EL12	x	x	0	-	-	n/a	-
CPACR_EL12	0	0	1	-	-	-	-
CPACR_EL12	0	1	1	-	n/a	-	-
CPACR_EL12	1	0	1	-	-	RW	RW
CPACR_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic CPACR\_EL1 or CPACR\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

## 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 [CPTR\\_EL2.TCPAC](#)==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 [CPTR\\_EL2.TCPAC](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [CPTR\\_EL3.TCPAC](#)==1, accesses to this register from EL1 and EL2 are trapped to EL3.

# CPTR\_EL2, Architectural Feature Trap Register (EL2)

The CPT<sub>R</sub>\_EL2 characteristics are:

## Purpose

Controls:

- Trapping to EL2 of access to [CPACR](#), [CPACR\\_EL1](#), trace functionality, and to SVE, Advanced SIMD and floating-point functionality.
- EL2 access to trace functionality, and to SVE, Advanced SIMD and floating-point functionality.

This register is part of the Virtualization registers functional group.

## Configuration

AArch64 System register CPT<sub>R</sub>\_EL2 is architecturally mapped to AArch32 System register [HCPTR](#).

If EL2 is not implemented, this register is RES0 from EL3.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CPT<sub>R</sub>\_EL2 is a 32-bit register.

## Field descriptions

The CPT<sub>R</sub>\_EL2 bit assignments are:

### When HCR\_EL2.E2H == 0:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<a href="#">TCPAC</a>	0	0	0	0	0	0	0	0	0	0	<a href="#">TTA</a>	0	0	0	0	0	0	1	1	0	<a href="#">TFP</a>	1	<a href="#">TZ</a>	1	1	1	1	1	1	1	1

This format applies in all ARMv8.0 implementations.

### TCPAC, bit [31]

Traps Non-secure EL1 accesses to [CPACR\\_EL1](#) or [CPACR](#) to EL2, from both Execution states.

TCPAC	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 accesses to <a href="#">CPACR_EL1</a> and <a href="#">CPACR</a> are trapped to EL2.

#### Note

[CPACR\\_EL1](#) and [CPACR](#) are not accessible at EL0.

### Bits [30:21]

Reserved, RES0.

### TTA, bit [20]

Traps Non-secure System register accesses to all implemented trace registers to EL2, from both Execution states.

TTA	Meaning
0	This control does not cause any instructions to be trapped.
1	Any attempt at EL2, or Non-secure EL0 or EL1, to execute a System register access to an implemented trace register is trapped to EL2, unless it is trapped by <a href="#">CPACR.NSTRCDIS</a> or <a href="#">CPACR_EL1.TTA</a> .

**Note**

- The ETMv4 architecture does not permit EL0 to access the trace registers. If the ARMv8-A architecture is implemented with an ETMv4 implementation, EL0 accesses to the trace registers are UNDEFINED, and any resulting exception is higher priority than an exception that would be generated because the value of [CPTR\\_EL2.TTA](#) is 1.
- EL2 does not provide traps on trace register accesses through the optional memory-mapped interface.

System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.

If System register access to the trace functionality is not supported, this bit is RES0.

**Bits [19:14]**

Reserved, RES0.

**Bits [13:12]**

Reserved, RES1.

**Bit [11]**

Reserved, RES0.

**TFP, bit [10]**

Traps Non-secure accesses to SVE, Advanced SIMD and floating-point functionality to EL2, from both Execution states.

TFP	Meaning
0	This control does not cause any instructions to be trapped.
1	Any attempt at EL2, or Non-secure EL0 or EL1, to execute an instruction that uses the registers associated with SVE, Advanced SIMD and floating-point execution is trapped to EL2, unless it is trapped by <a href="#">CPACR.cp10</a> , <a href="#">CPACR_EL1.FPEN</a> , and if SVE is implemented, <a href="#">CPACR_EL1.ZEN</a> , or <a href="#">CPTR_EL2.TZ</a> .

**Bit [9]**

Reserved, RES1.

**TZ, bit [8]**

In ARMv8.2:

Present only if SVE is implemented.

Traps Non-secure execution at EL2, EL1, or EL0 of SVE instructions and instructions that access SVE System registers to EL2. Defined values are:

TZ	Meaning
0	This control does not cause any instruction to be trapped.
1	This control causes these instructions to be trapped, unless <a href="#">HCR_EL2.TGE</a> is 0 and they are trapped by <a href="#">CPACR_EL1</a> .

If SVE is not implemented, this field is RES1.

In ARMv8.1 and ARMv8.0:

Reserved, RES1.

#### Bits [7:0]

Reserved, RES1.

#### When HCR\_EL2.E2H == 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
	TCPAC	0	0	TTA	0	0	0	0	0	0	FPEN	0	0	ZEN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

#### TCPAC, bit [31]

When [HCR\\_EL2.TGE](#) is 0, traps Non-secure EL1 accesses to [CPACR\\_EL1](#) and [CPACR](#) to EL2, from both Execution states.

TCPAC	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 accesses to <a href="#">CPACR_EL1</a> and <a href="#">CPACR</a> are trapped to EL2.

When [HCR\\_EL2.TGE](#) is 1, this control does not cause any instructions to be trapped.

#### Note

[CPACR\\_EL1](#) and [CPACR](#) are not accessible at EL0.

#### Bits [30:29]

Reserved, RES0.

#### TTA, bit [28]

Traps Non-secure System register accesses to all implemented trace registers to EL2, from both Execution states.

TTA	Meaning
0	This control does not cause any instructions to be trapped.
1	Any attempt at EL2, or Non-secure EL0 or EL1, to execute a System register access to an implemented trace register is trapped to EL2, unless <a href="#">HCR_EL2.TGE</a> is 0 and it is trapped by <a href="#">CPACR.NSTRCDIS</a> or <a href="#">CPACR_EL1.TTA</a> . When <a href="#">HCR_EL2.TGE</a> is 1, any attempt at EL2, or Non-secure EL0, to execute a System register access to an implemented trace register is trapped to EL2.

#### Note

- The ETMv4 architecture does not permit EL0 to access the trace registers. If the ARMv8-A architecture is implemented with an ETMv4 implementation, EL0 accesses to the trace registers are UNDEFINED, and any resulting exception is higher priority than an exception that would be generated because the value of [CPTR\\_EL2.TTA](#) is 1.
- EL2 does not provide traps on trace register accesses through the optional memory-mapped interface.

System register accesses to the trace registers can have side-effects. When a System register access is trapped, any side-effects that are normally associated with the access do not occur before the exception is taken.

If System register access to the trace functionality is not supported, this bit is RES0.

#### Bits [27:22]

Reserved, RES0.

**FPEN, bits [21:20]**

Traps EL2, Non-secure EL0 and, when [HCR\\_EL2.TGE](#) is 0, Non-secure EL1 accesses to the SVE, Advanced SIMD and floating-point registers to EL2, from both Execution states.

FPEN	Meaning
00	This control causes any instructions at Non-secure EL0, EL1, or EL2 that use the registers associated with SVE, Advanced SIMD and floating-point execution to be trapped, unless <a href="#">HCR_EL2.TGE</a> is 0 and they are trapped by <a href="#">CPACR.cp10</a> , <a href="#">CPACR_EL1.FPEN</a> , <a href="#">CPACR_EL1.ZEN</a> , or <a href="#">CPTR_EL2.ZEN</a> .
01	When <a href="#">HCR_EL2.TGE</a> is 0, this control does not cause any instructions to be trapped. When <a href="#">HCR_EL2.TGE</a> is 1, this control causes instructions at Non-secure EL0 that use the registers associated with SVE, Advanced SIMD and floating-point execution to be trapped, unless they are trapped by <a href="#">CPTR_EL2.ZEN</a> .
10	This control causes any instructions at Non-secure EL0, EL1, or EL2 that use the registers associated with SVE, Advanced SIMD and floating-point execution to be trapped, unless <a href="#">HCR_EL2.TGE</a> is 0 and they are trapped by <a href="#">CPACR.cp10</a> , <a href="#">CPACR_EL1.FPEN</a> , <a href="#">CPACR_EL1.ZEN</a> , or <a href="#">CPTR_EL2.ZEN</a> .
11	This control does not cause any instructions to be trapped.

Writes to [MVFR0](#), [MVFR1](#), and [MVFR2](#) from EL1 or higher are CONSTRAINED UNPREDICTABLE and whether these accesses can be trapped by this control depends on implemented CONSTRAINED UNPREDICTABLE behavior.

**Note**

- Attempts to write to the FPSID count as use of the registers for accesses from EL1 or higher.
- Accesses from EL0 to [FPSID](#), [MVFR0](#), [MVFR1](#), [MVFR2](#), and [FPEXC](#) are UNDEFINED, and any resulting exception is higher priority than an exception that would be generated because the value of [CPTR\\_EL2.FPEN](#) is not 11.

**Bits [19:18]**

Reserved, RES0.

**ZEN, bits [17:16]**

In ARMv8.2:

Present only if SVE is implemented.

Traps Non-secure execution at EL2, EL1, and EL0 of SVE instructions or instructions that access SVE System registers to EL2.

Defined values are:

ZEN	Meaning
00	This control causes Non-secure execution at EL2, EL1, and EL0 of these instructions to be trapped, unless <a href="#">HCR_EL2.TGE</a> is 0 and they are trapped by <a href="#">CPACR_EL1</a> .
01	When <a href="#">HCR_EL2.TGE</a> is 0, this control does not cause any instruction to be trapped. When <a href="#">HCR_EL2.TGE</a> is 1, this control causes these instructions executed at Non-secure EL0 to be trapped, but does not cause any instruction at EL2 to be trapped.
10	This control causes Non-secure execution at EL2, EL1, and EL0 of these instructions to be trapped, unless <a href="#">HCR_EL2.TGE</a> is 0 and they are trapped by <a href="#">CPACR_EL1</a> .
11	This control does not cause any instruction to be trapped.

If SVE is not implemented, this field is RES0.

In ARMv8.1:

Reserved, RES0.

**Bits [15:0]**

Reserved, RES0.

## Accessing the CPTR\_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
CPTR_EL2	11	100	0001	0001	010
CPACR_EL1	11	000	0001	0000	010

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
CPTR_EL2	x	x	0	-	-	n/a	RW
CPTR_EL2	0	0	1	-	-	RW	RW
CPTR_EL2	0	1	1	-	n/a	RW	RW
CPTR_EL2	1	0	1	-	-	RW	RW
CPTR_EL2	1	1	1	-	n/a	RW	RW
CPACR_EL1	x	x	0	-	<a href="#">CPACR_EL1</a>	n/a	<a href="#">CPACR_EL1</a>
CPACR_EL1	0	0	1	-	<a href="#">CPACR_EL1</a>	<a href="#">CPACR_EL1</a>	<a href="#">CPACR_EL1</a>
CPACR_EL1	0	1	1	-	n/a	<a href="#">CPACR_EL1</a>	<a href="#">CPACR_EL1</a>
CPACR_EL1	1	0	1	-	<a href="#">CPACR_EL1</a>	RW	<a href="#">CPACR_EL1</a>
CPACR_EL1	1	1	1	-	n/a	RW	<a href="#">CPACR_EL1</a>

## 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 :

- If [CPTR\\_EL3](#).TCPAC==1, accesses to this register from EL2 are trapped to EL3.

# CPTR\_EL3, Architectural Feature Trap Register (EL3)

The CPTR\_EL3 characteristics are:

## Purpose

Controls trapping to EL3 of access to [CPACR\\_EL1](#), [CPTR\\_EL2](#), trace functionality and registers associated with SVE, Advanced SIMD and floating-point execution. Also controls EL3 access to trace functionality and registers associated with SVE, Advanced SIMD and floating-point execution.

This register is part of the Security registers functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

CPTR\_EL3 is a 32-bit register.

## Field descriptions

The CPTR\_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
<a href="#">TCPAC</a>	0	0	0	0	0	0	0	0	0	0	<a href="#">TTA</a>	0	0	0	0	0	0	0	0	0	<a href="#">TFP</a>	0	<a href="#">EZ</a>	0	0	0	0	0	0	0	0

### TCPAC, bit [31]

Traps all of the following to EL3, from both Security states and both Execution states.

- EL2 accesses to the [CPTR\\_EL2](#) or [HCPTR](#).
- EL2 and EL1 accesses to the [CPACR\\_EL1](#) or [CPACR](#).

When CPTR\_EL3.TCPAC is:

TCPAC	Meaning
0	This control does not cause any instructions to be trapped.
1	EL2 accesses to the <a href="#">CPTR_EL2</a> or <a href="#">HCPTR</a> , and EL2 and EL1 accesses to the <a href="#">CPACR_EL1</a> or <a href="#">CPACR</a> , are trapped to EL3, unless they are trapped by <a href="#">CPTR_EL2</a> .TCPAC.

### Bits [30:21]

Reserved, RES0.

### TTA, bit [20]

Traps System register accesses to the trace registers, from all Exception levels, both Security states, and both Execution states, to EL3.

TTA	Meaning
0	This control does not cause any instructions to be trapped.
1	Any System register access to the trace registers is trapped to EL3, subject to the exception prioritization rules, unless it is trapped by <a href="#">CPACR</a> .NSTRCDIS, <a href="#">CPACR_EL1</a> .TTA or <a href="#">CPTR_EL2</a> .TTA.

If System register access to trace functionality is not supported, this bit is RES0.



**Bits [19:11]**

Reserved, RES0.

**TFP, bit [10]**

Traps all accesses to SVE, Advanced SIMD and floating-point functionality, from all Exception levels, both Security states, and both Execution states, to EL3. Defined values are:

TFP	Meaning
0	This control does not cause any instructions to be trapped.
1	Any attempt at any Exception level to execute an instruction that uses the registers associated with SVE, Advanced SIMD and floating-point is trapped to EL3, subject to the exception prioritization rules, unless it is trapped by <a href="#">CPACR.cp10</a> , <a href="#">CPACR_EL1.FPEN</a> , <a href="#">CPTR_EL2.TFP</a> , and if SVE is implemented, <a href="#">CPACR_EL1.ZEN</a> , <a href="#">CPTR_EL2.FPEN</a> , <a href="#">CPTR_EL2.ZEN</a> , or <a href="#">CPTR_EL2.TZ</a> .

**Bit [9]**

Reserved, RES0.

**EZ, bit [8]****In ARMv8.2:**

Present only if SVE is implemented.

Traps all accesses to SVE functionality and registers from all Exception levels, and both Security states, to EL3. Defined values are:

EZ	Meaning
0	This control causes these instructions executed at any Exception level to be trapped unless they are trapped by <a href="#">CPTR_EL2</a> or <a href="#">CPACR_EL1</a> .
1	This control does not cause any instruction to be trapped.

If SVE is not implemented, this field is RES0.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**Bits [7:0]**

Reserved, RES0.

**Accessing the CPTR\_EL3**

This register can be read using MRS with the following syntax:

MRS &lt;Xt&gt;, &lt;systemreg&gt;

This register can be written using MSR (register) with the following syntax:

MSR &lt;systemreg&gt;, &lt;Xt&gt;

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
CPTR_EL3	11	110	0001	0001	010

**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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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	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 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.

# CTR\_EL0, Cache Type Register

The CTR\_EL0 characteristics are:

## Purpose

Provides information about the architecture of the caches.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register CTR\_EL0 is architecturally mapped to AArch32 System register [CTR](#).

## Attributes

CTR\_EL0 is a 32-bit register.

## Field descriptions

The CTR\_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
1	0	0	0	CWG				ERG				DminLine				L1Ip		0	0	0	0	0	0	0	0	0	0	IminLine			

### Bit [31]

Reserved, RES1.

### Bits [30:28]

Reserved, RES0.

### CWG, bits [27:24]

Cache writeback granule. Log<sub>2</sub> of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified.

A value of 0b0000 indicates that this register does not provide Cache writeback granule information and either:

- The architectural maximum of 512 words (2KB) must be assumed.
- The Cache writeback granule can be determined from maximum cache line size encoded in the Cache Size ID Registers.

Values greater than 0b1001 are reserved.

ARM recommends that an implementation that does not support cache write-back implements this field as 0b0001. This applies, for example, to an implementation that supports only write-through caches.

### ERG, bits [23:20]

Exclusives reservation granule. Log<sub>2</sub> of the number of words of the maximum size of the reservation granule that has been implemented for the Load-Exclusive and Store-Exclusive instructions.

A value of 0b0000 indicates that this register does not provide Exclusives reservation granule information and the architectural maximum of 512 words (2KB) must be assumed.

Values greater than 0b1001 are reserved.

**DminLine, bits [19:16]**

Log<sub>2</sub> of the number of words in the smallest cache line of all the data caches and unified caches that are controlled by the PE.

**L1Ip, bits [15:14]**

Level 1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache. Possible values of this field are:

L1Ip	Meaning
00	VMID aware Physical Index, Physical tag (VPIPT)
01	ASID-tagged Virtual Index, Virtual Tag (AIVIVT)
10	Virtual Index, Physical Tag (VIPT)
11	Physical Index, Physical Tag (PIPT)

The value 0b01 is reserved in ARMv8.

The value 0b00 is permitted only in an implementation that includes ARMv8.2-PIPTV, otherwise the value is reserved.

**Bits [13:4]**

Reserved, RES0.

**IminLine, bits [3:0]**

Log<sub>2</sub> of the number of words in the smallest cache line of all the instruction caches that are controlled by the PE.

**Accessing the CTR\_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
CTR_EL0	11	011	0000	0000	001

**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<sup>®</sup> 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 [SCTLR\\_EL1.UCT](#)==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 [HCR\\_EL2](#).TID2==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 [HCR\\_EL2](#).TID2==1, Non-secure read accesses to this register from EL0 and EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CurrentEL, Current Exception Level

The CurrentEL characteristics are:

## Purpose

Holds the current Exception level.

This register is part of the Process state registers functional group.

## Configuration

There are no configuration notes.

## Attributes

CurrentEL is a 32-bit register.

## Field descriptions

The CurrentEL 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	EL		0	0

### Bits [31:4]

Reserved, RES0.

### EL, bits [3:2]

Current Exception level. Possible values of this field are:

EL	Meaning
00	EL0
01	EL1
10	EL2
11	EL3

When this register has an architecturally-defined reset value, this field resets to an architecturally UNKNOWN value.

### Bits [1:0]

Reserved, RES0.

## Accessing the CurrentEL

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
CurrentEL	11	000	0100	0010	010



## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DACR32\_EL2, Domain Access Control Register

The DACR32\_EL2 characteristics are:

## Purpose

Allows access to the AArch32 [DACR](#) register from AArch64 state only. Its value has no effect on execution in AArch64 state.

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch64 System register DACR32\_EL2 is architecturally mapped to AArch32 System register [DACR](#).

If EL1 does not support AArch32, this register is UNDEFINED.

If EL2 is not implemented but EL3 is implemented, and EL1 is capable of using AArch32, then this register is not RES0.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

DACR32\_EL2 is a 32-bit register.

## Field descriptions

The DACR32\_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
D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0																

**D<n>, bits [2n+1:2n], for n = 0 to 15**

Domain n access permission, where n = 0 to 15. Permitted values are:

D<n>	Meaning
00	No access. Any access to the domain generates a Domain fault.
01	Client. Accesses are checked against the permission bits in the translation tables.
11	Manager. Accesses are not checked against the permission bits in the translation tables.

The value 10 is reserved.

## Accessing the DACR32\_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
DACR32_EL2	11	100	0011	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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DAIF, Interrupt Mask Bits

The DAIF characteristics are:

## Purpose

Allows access to the interrupt mask bits.

This register is part of the Process state registers functional group.

## Configuration

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 AArch64. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

DAIF is a 32-bit register.

## Field descriptions

The DAIF 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	D	A	I	F	0	0	0	0	0	0

### Bits [31:10]

Reserved, RES0.

### D, bit [9]

Process state D mask. The possible values of this bit are:

D	Meaning
0	Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are not masked.
1	Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are masked.

When the target Exception level of the debug exception is higher than the current Exception level, the exception is not masked by this bit.

When this register has an architecturally-defined reset value, this field resets to 1.

### A, bit [8]

SError interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

When this register has an architecturally-defined reset value, this field resets to 1.

### I, bit [7]

IRQ mask bit. The possible values of this bit are:

<b>I</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

When this register has an architecturally-defined reset value, this field resets to 1.

## F, bit [6]

FIQ mask bit. The possible values of this bit are:

<b>F</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

When this register has an architecturally-defined reset value, this field resets to 1.

## Bits [5:0]

Reserved, RES0.

# Accessing the DAIF

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:

<b>&lt;systemreg&gt;</b>	<b>op0</b>	<b>op1</b>	<b>CRn</b>	<b>CRm</b>	<b>op2</b>
DAIF	11	011	0100	0010	001

This register can be modified using MSR (immediate) with the following syntax:

```
MSR <pstatefield>, <imm>
```

This syntax uses the following encoding in the System instruction encoding space:

<b>&lt;pstatefield&gt;</b>	<b>op0</b>	<b>op1</b>	<b>CRn</b>	<b>op2</b>
DAIFSet	00	011	0100	110
DAIFClr	00	011	0100	111

## Accessibility

The register is accessible as follows:

<b>Control</b>			<b>Accessibility</b>			
<b>E2H</b>	<b>TGE</b>	<b>NS</b>	<b>EL0</b>	<b>EL1</b>	<b>EL2</b>	<b>EL3</b>
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 [SCTLR\\_EL1](#).UMA==0, accesses to this register from EL0 are trapped to EL1.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DBGAUTHSTATUS\_EL1, Debug Authentication Status register

The DBGAUTHSTATUS\_EL1 characteristics are:

## Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register DBGAUTHSTATUS\_EL1 is architecturally mapped to AArch32 System register [DBGAUTHSTATUS](#).

AArch64 System register DBGAUTHSTATUS\_EL1 is architecturally mapped to External register [DBGAUTHSTATUS\\_EL1](#).

## Attributes

DBGAUTHSTATUS\_EL1 is a 32-bit register.

## Field descriptions

The DBGAUTHSTATUS\_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	<a href="#">SNID</a>	<a href="#">SID</a>	<a href="#">NSNID</a>	<a href="#">NSID</a>				

### Bits [31:8]

Reserved, RES0.

### SNID, bits [7:6]

Secure non-invasive debug. Possible values of this field are:

SNID	Meaning
00	Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state.
10	Implemented and disabled. ExternalSecureNoninvasiveDebugEnabled() == FALSE.
11	Implemented and enabled. ExternalSecureNoninvasiveDebugEnabled() == TRUE.

Other values are reserved.

### SID, bits [5:4]

Secure invasive debug. Possible values of this field are:

SID	Meaning
00	Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state.
10	Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE.
11	Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.

Other values are reserved.

**NSNID, bits [3:2]**

Non-secure non-invasive debug. Possible values of this field are:

NSNID	Meaning
00	Not implemented. EL3 is not implemented and the implemented Security state is Secure state.
10	Implemented and disabled. ExternalNoninvasiveDebugEnabled() == FALSE.
11	Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE.

Other values are reserved.

**NSID, bits [1:0]**

Non-secure invasive debug. Possible values of this field are:

NSID	Meaning
00	Not implemented. EL3 is not implemented and the implemented Security state is Secure state.
10	Implemented and disabled. ExternalInvasiveDebugEnabled() == FALSE.
11	Implemented and enabled. ExternalInvasiveDebugEnabled() == TRUE.

Other values are reserved.

**Accessing the DBGAUTHSTATUS\_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
DBGAUTHSTATUS_EL1	10	000	0111	1110	110

**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 :

- If [MDCR\\_EL2.TDA==1](#), Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3.TDA==1](#), read accesses to this register from EL1 and EL2 are trapped to EL3.





# 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 <a href="#">CONTEXTIDR_EL1</a> match (introduced in ARMv8.1).
0111	Linked <a href="#">CONTEXTIDR_EL1</a> 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 <a href="#">CONTEXTIDR_EL2</a> match (introduced in ARMv8.1).
1101	Linked <a href="#">CONTEXTIDR_EL2</a> 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	<a href="#">DBGBVR&lt;n&gt;_EL1</a>	Use for T32 instructions.
1100	<a href="#">DBGBVR&lt;n&gt;_EL1+2</a>	Use for T32 instructions.
1111	<a href="#">DBGBVR&lt;n&gt;_EL1</a>	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, accesses to this register from EL1, EL2, and EL3 are trapped to Debug 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.

# 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 ARMv8.2:

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 ARMv8.2 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 ARMv8.2 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, accesses to this register from EL1, EL2, and EL3 are trapped to Debug 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DBGCLAIMCLR\_EL1, Debug Claim Tag Clear register

The DBGCLAIMCLR\_EL1 characteristics are:

## Purpose

Used by software to read the values of the CLAIM tag bits, and to clear these bits to 0.

The architecture does not define any functionality for the CLAIM tag bits.

---

### Note

CLAIM tags are typically used for communication between the debugger and target software.

---

Used in conjunction with the [DBGCLAIMSET\\_EL1](#) register.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register DBGCLAIMCLR\_EL1 is architecturally mapped to AArch32 System register [DBGCLAIMCLR](#).

AArch64 System register DBGCLAIMCLR\_EL1 is architecturally mapped to External register [DBGCLAIMCLR\\_EL1](#).

An implementation must include 8 CLAIM tag bits.

This register is in the Cold reset domain. See the CLAIM field description for the effect of a Cold reset on the value returned by this register. This register is not affected by a Warm reset.

## Attributes

DBGCLAIMCLR\_EL1 is a 32-bit register.

## Field descriptions

The DBGCLAIMCLR\_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								

CLAIM

### Bits [31:8]

Reserved, RAZ/SBZ. Software can rely on these bits reading as zero, and must use a should-be-zero policy on writes. Implementations must ignore writes.

### CLAIM, bits [7:0]

Read or clear CLAIM tag bits. Reading this field returns the current value of the CLAIM tag bits.

Writing a 1 to one of these bits clears the corresponding CLAIM tag bit to 0. This is an indirect write to the CLAIM tag bits. A single write operation can clear multiple CLAIM tag bits to 0.

Writing 0 to one of these bits has no effect.

A cold reset clears the CLAIM tag bits to 0.

## Accessing the DBGCLAIMCLR\_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
DBGCLAIMCLR_EL1	10	000	0111	1001	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.

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.

# DBGCLAIMSET\_EL1, Debug Claim Tag Set register

The DBGCLAIMSET\_EL1 characteristics are:

## Purpose

Used by software to set the CLAIM tag bits to 1.

The architecture does not define any functionality for the CLAIM tag bits.

---

### Note

CLAIM tags are typically used for communication between the debugger and target software.

---

Used in conjunction with the [DBGCLAIMCLR\\_EL1](#) register.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register DBGCLAIMSET\_EL1 is architecturally mapped to AArch32 System register [DBGCLAIMSET](#).

AArch64 System register DBGCLAIMSET\_EL1 is architecturally mapped to External register [DBGCLAIMSET\\_EL1](#).

An implementation must include 8 CLAIM tag bits.

## Attributes

DBGCLAIMSET\_EL1 is a 32-bit register.

## Field descriptions

The DBGCLAIMSET\_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								

CLAIM

### Bits [31:8]

Reserved, RAZ/SBZ. Software can rely on these bits reading as zero, and must use a should-be-zero policy on writes. Implementations must ignore writes.

### CLAIM, bits [7:0]

Set CLAIM tag bits. RAO.

Writing a 1 to one of these bits sets the corresponding CLAIM tag bit to 1. This is an indirect write to the CLAIM tag bits. A single write operation can set multiple CLAIM tag bits to 1.

Writing 0 to one of these bits has no effect.

A cold reset clears the CLAIM tag bits to 0.

## Accessing the DBGCLAIMSET\_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
DBGCLAIMSET_EL1	10	000	0111	1000	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.

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.

# DBGDTRRX\_EL0, Debug Data Transfer Register, Receive

The DBGDTRRX\_EL0 characteristics are:

## Purpose

Transfers data from an external debugger to the PE. For example, it is used by a debugger transferring commands and data to a debug target. It is a component of the Debug Communications Channel.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register DBGDTRRX\_EL0 is architecturally mapped to AArch32 System register [DBGDTRRXint](#).

AArch64 System register DBGDTRRX\_EL0 is architecturally mapped to External register [DBGDTRRX\\_EL0](#).

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

DBGDTRRX\_EL0 is a 32-bit register.

## Field descriptions

The DBGDTRRX\_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
Update DTRRX																															

### Bits [31:0]

Update DTRRX.

If RXfull is set to 1, then reads of this register return the last value written to DTRRX and clear RXfull to 0.

For the full behavior of the Debug Communications Channel, see 'The Debug Communication Channel and Instruction Transfer Register' in the ARM ARM, chapter H4.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the DBGDTRRX\_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
DBGDTRRX_EL0	10	011	0000	0101	000

## 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<sup>®</sup> 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 [MDSCR\\_EL1](#).TDCC==1, 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](#).TDA==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](#).TDA==1, read accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DBGDTRTX\_EL0, Debug Data Transfer Register, Transmit

The DBGDTRTX\_EL0 characteristics are:

## Purpose

Transfers data from the PE to an external debugger. For example, it is used by a debug target to transfer data to the debugger. It is a component of the Debug Communication Channel.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register DBGDTRTX\_EL0 is architecturally mapped to AArch32 System register [DBGDTRTXint](#).

AArch64 System register DBGDTRTX\_EL0 is architecturally mapped to External register [DBGDTRTX\\_EL0](#).

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

DBGDTRTX\_EL0 is a 32-bit register.

## Field descriptions

The DBGDTRTX\_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
Return DTRTX																															

### Bits [31:0]

Return DTRTX.

If TXfull is set to 0, then writes of this register update the value in DTRTX and set TXfull to 1.

For the full behavior of the Debug Communications Channel, see 'The Debug Communication Channel and Instruction Transfer Register' in the ARM ARM, chapter H4.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the DBGDTRTX\_EL0

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
DBGDTRTX_EL0	10	011	0000	0101	000

## 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 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 [MDSCR\\_EL1.TDCC](#)==1, write 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.TDA](#)==1, Non-secure write accesses to this register from EL0 and EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3.TDA](#)==1, write accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DBGDTR\_EL0, Debug Data Transfer Register, half-duplex

The DBGDTR\_EL0 characteristics are:

## Purpose

Transfers 64 bits of data between the PE and an external debugger. Can transfer both ways using only a single register.

This register is part of the Debug registers functional group.

## Configuration

There are no configuration notes.

## Attributes

DBGDTR\_EL0 is a 64-bit register.

## Field descriptions

The DBGDTR\_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
																HighWord															
																LowWord															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### HighWord, bits [63:32]

Writes to this register set DTRRX to the value in this field and do not change RXfull.

Reads from this register return the value of DTRTX and do not change TXfull.

### LowWord, bits [31:0]

Writes to this register set DTRTX to the value in this field and set TXfull to 1.

Reads from this register return the value of DTRRX and clear RXfull to 0.

## Accessing the DBGDTR\_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
DBGDTR_EL0	10	011	0000	0100	000

## 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<sup>®</sup> 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 [MDSCR\\_EL1](#).TDCC==1, 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](#).TDA==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](#).TDA==1, accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DBGPRCR\_EL1, Debug Power Control Register

The DBGPRCR\_EL1 characteristics are:

## Purpose

- Controls behavior of the PE on powerdown request.
- This register is part of the Debug registers functional group.

## Configuration

- AArch64 System register DBGPRCR\_EL1 is architecturally mapped to AArch32 System register [DBGPRCR](#).
- Bit [0] of this register is mapped to [EDPRCR](#).CORENPDRQ, bit [0] of the external view of this register.
- The other bits in these registers are not mapped to each other.
- 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

DBGPRCR\_EL1 is a 32-bit register.

## Field descriptions

The DBGPRCR\_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	0	0	<a href="#">CORENPDRQ</a>

### Bits [31:1]

Reserved, RES0.

### CORENPDRQ, bit [0]

Core no powerdown request. Requests emulation of powerdown. Possible values of this bit are:

CORENPDRQ	Meaning
0	If the system responds to a powerdown request, it powers down Core power domain.
1	If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain.

Writes to this bit are permitted regardless of the state of the IMPLEMENTATION DEFINED authentication interface. This means that a debugger can request Core no powerdown regardless of whether invasive debug is permitted.

It is IMPLEMENTATION DEFINED whether this bit is reset to the value of [EDPRCR](#).COREPURQ on exit from an IMPLEMENTATION DEFINED software-visible retention state.

## Accessing the DBGPRCR\_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
DBGPRCR_EL1	10	000	0001	0100	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.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 :

- If [MDCR\\_EL2](#).TDOSA==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDOSA==1, accesses to this register from EL1 and EL2 are trapped to EL3.

# DBGVCR32\_EL2, Debug Vector Catch Register

The DBGVCR32\_EL2 characteristics are:

## Purpose

Allows access to the AArch32 register [DBGVCR](#) from AArch64 state only. Its value has no effect on execution in AArch64 state.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register DBGVCR32\_EL2 is architecturally mapped to AArch32 System register [DBGVCR](#).

If EL1 does not support AArch32, this register is UNDEFINED.

If EL2 is not implemented but EL3 is implemented, and EL1 is capable of using AArch32, then this register is not RES0.

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

DBGVCR32\_EL2 is a 32-bit register.

## Field descriptions

The DBGVCR32\_EL2 bit assignments are:

### When EL3 implemented and using AArch64:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NSF	NSI	0	NSD	NSP	NSS	NSU	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SF	SI	0	SD	SP	SS	SU	0	

#### NSF, bit [31]

FIQ vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 1C$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### NSI, bit [30]

IRQ vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 18$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Bit [29]

Reserved, RES0.

#### NSD, bit [28]

Data Abort vector catch enable in Non-secure state.



The exception vector offset is  $0 \times 10$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### NSP, bit [27]

Prefetch Abort vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 0C$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### NSS, bit [26]

Supervisor Call (SVC) vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 08$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### NSU, bit [25]

Undefined Instruction vector catch enable in Non-secure state.

The exception vector offset is  $0 \times 04$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Bits [24:8]

Reserved, RES0.

#### SF, bit [7]

FIQ vector catch enable in Secure state.

The exception vector offset is  $0 \times 1C$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### SI, bit [6]

IRQ vector catch enable in Secure state.

The exception vector offset is  $0 \times 18$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Bit [5]

Reserved, RES0.

#### SD, bit [4]

Data Abort vector catch enable in Secure state.

The exception vector offset is  $0 \times 10$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**SP, bit [3]**

Prefetch Abort vector catch enable in Secure state.

The exception vector offset is  $0 \times 0C$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**SS, bit [2]**

Supervisor Call (SVC) vector catch enable in Secure state.

The exception vector offset is  $0 \times 08$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**SU, bit [1]**

Undefined Instruction vector catch enable in Secure state.

The exception vector offset is  $0 \times 04$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Bit [0]**

Reserved, RES0.

**When EL3 not 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	0	0	0	0	0	0	0	0	F	I	0	D	P	S	U	0

**Bits [31:8]**

Reserved, RES0.

**F, bit [7]**

FIQ vector catch enable.

The exception vector offset is  $0 \times 1C$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**I, bit [6]**

IRQ vector catch enable.

The exception vector offset is  $0 \times 18$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Bit [5]**

Reserved, RES0.

**D, bit [4]**

Data Abort vector catch enable.

The exception vector offset is  $0 \times 10$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### P, bit [3]

Prefetch Abort vector catch enable.

The exception vector offset  $0 \times 0C$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### S, bit [2]

Supervisor Call (SVC) vector catch enable.

The exception vector offset is  $0 \times 08$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### U, bit [1]

Undefined Instruction vector catch enable.

The exception vector offset is  $0 \times 04$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### Bit [0]

Reserved, RES0.

## Accessing the DBGVCR32\_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
DBGVCR32_EL2	10	100	0000	0111	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 EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDA==1, accesses to this register from EL2 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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			MASK			0	0	0	WT		LBN			SSC	HMC					BAS					LSC		PAC		E

### 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 <a href="#">DBGWVR&lt;n&gt;_EL1</a>
xxxxxx1x	Match byte at <a href="#">DBGWVR&lt;n&gt;_EL1+1</a>
xxxxx1xx	Match byte at <a href="#">DBGWVR&lt;n&gt;_EL1+2</a>
xxxx1xxx	Match byte at <a href="#">DBGWVR&lt;n&gt;_EL1+3</a>

In cases where [DBGWVR<n>\\_EL1](#) addresses a double-word:

BAS	Description, if <a href="#">DBGWVR&lt;n&gt;_EL1[2]</a> == 0
xxx1xxxx	Match byte at <a href="#">DBGWVR&lt;n&gt;_EL1+4</a>
xx1xxxxx	Match byte at <a href="#">DBGWVR&lt;n&gt;_EL1+5</a>
x1xxxxxx	Match byte at <a href="#">DBGWVR&lt;n&gt;_EL1+6</a>
1xxxxxxx	Match byte at <a href="#">DBGWVR&lt;n&gt;_EL1+7</a>

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, accesses to this register from EL1, EL2, and EL3 are trapped to Debug 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# 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 ARMv8.2:

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, accesses to this register from EL1, EL2, and EL3 are trapped to Debug 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.



# DCZID\_EL0, Data Cache Zero ID register

The DCZID\_EL0 characteristics are:

## Purpose

Indicates the block size that is written with byte values of 0 by the [DC ZVA](#) (Data Cache Zero by Address) system instruction.

This register is part of the Identification registers functional group.

## Configuration

There are no configuration notes.

## Attributes

DCZID\_EL0 is a 32-bit register.

## Field descriptions

The DCZID\_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	DZP		BS		

### Bits [31:5]

Reserved, RES0.

### DZP, bit [4]

Data Zero prohibited. Permitted values are:

DZP	Meaning
0	<a href="#">DC ZVA</a> instruction is permitted.
1	<a href="#">DC ZVA</a> instruction is prohibited.

The value read from this field is governed by the access state and the values of the [HCR\\_EL2](#).TDZ and [SCTLR\\_EL1](#).DZE bits.

### BS, bits [3:0]

Log<sub>2</sub> of the block size in words. The maximum size supported is 2KB (value == 9).

## Accessing the DCZID\_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
DCZID_EL0	11	011	0000	0000	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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DLR\_EL0, Debug Link Register

The DLR\_EL0 characteristics are:

## Purpose

In Debug state, holds the address to restart from.

This register is part of:

- The Debug registers functional group.
- The Special-purpose registers functional group.

## Configuration

AArch64 System register DLR\_EL0 bits [31:0] are architecturally mapped to AArch32 System register [DLR](#).

## Attributes

DLR\_EL0 is a 64-bit register.

## Field descriptions

The DLR\_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
Restart address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Restart address																															

### Bits [63:0]

Restart address.

## Accessing the DLR\_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
DLR_EL0	11	011	0100	0101	001

## Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	RW	RW	n/a	RW

# DLR\_EL0, Debug Link Register

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.

Access to this register is from Debug state only. During normal execution this register is unallocated.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DSPSR\_EL0, Debug Saved Program Status Register

The DSPSR\_EL0 characteristics are:

## Purpose

Holds the saved process state on entry to Debug state.

This register is part of:

- The Debug registers functional group.
- The Special-purpose registers functional group.

## Configuration

AArch64 System register DSPSR\_EL0 is architecturally mapped to AArch32 System register [DSPSR](#).

## Attributes

DSPSR\_EL0 is a 32-bit register.

## Field descriptions

The DSPSR\_EL0 bit assignments are:

### When exiting Debug state to AArch32:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	Z	C	V	Q	IT[1:0]	J	0	PAN	SS	IL		GE		IT[7:2]				E	A	I	F	T	M[4]		M[3:0]						

#### N, bit [31]

Copied to [CPSR](#).N on exiting Debug state.

#### Z, bit [30]

Copied to [CPSR](#).Z on exiting Debug state.

#### C, bit [29]

Copied to [CPSR](#).C on exiting Debug state.

#### V, bit [28]

Copied to [CPSR](#).V on exiting Debug state.

#### Q, bit [27]

Cumulative saturation bit. Set to 1 to indicate that overflow or saturation occurred in some instructions.

#### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.



**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR](#).PAN on entering Debug state, and copied to [CPSR](#).PAN on exiting Debug state.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**SS, bit [21]**

Software step. Shows the value of PSTATE.SS immediately before Debug state was entered.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before Debug state was entered.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

<b>E</b>	<b>Meaning</b>
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTLR.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

SError interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

IRQ mask bit. The possible values of this bit are:

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

FIQ mask bit. The possible values of this bit are:

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the Debug state entry was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that Debug state was entered from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that Debug state was entered from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor
0b0111	Abort
0b1010	Hyp
0b1011	Undefined
0b1111	System

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.

## When entering Debug state from AArch64 and exiting Debug state to AArch64:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	Z	C	V	0	0	0	0	UAO	PAN	SS	IL	0	0	0	0	0	0	0	0	0	0	0	D	A	I	F	0	M[4]	M[3:0]		

### N, bit [31]

Set to the value of the N condition flag on entering Debug state, and copied to the N condition flag on exiting Debug state.

### Z, bit [30]

Set to the value of the Z condition flag on entering Debug state, and copied to the Z condition flag on exiting Debug state.

### C, bit [29]

Set to the value of the C condition flag on entering Debug state, and copied to the C condition flag on exiting Debug state.

### V, bit [28]

Set to the value of the V condition flag on entering Debug state, and copied to the V condition flag on exiting Debug state.

### Bits [27:24]

Reserved, RES0.

### UAO, bit [23]

#### In ARMv8.2:

When ARMv8.2-UAO is implemented, set to the value of PSTATE.UAO on entering Debug state, and copied to PSTATE.UAO on exiting Debug state.

When ARMv8.2-UAO is not implemented, this bit is RES0.

#### In ARMv8.1 and ARMv8.0:

Reserved, RES0.

### PAN, bit [22]

#### In ARMv8.2 and ARMv8.1:

When ARMv8.1-PAN is implemented, set to the value of PSTATE.PAN on entering Debug state, and copied to PSTATE.PAN on exiting Debug state.

When ARMv8.1-PAN is not implemented, this bit is RES0.

#### In ARMv8.0:

Reserved, RES0.

### SS, bit [21]

Software step. Shows the value of PSTATE.SS immediately before Debug state was entered.

### IL, bit [20]

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before Debug state was entered.

**Bits [19:10]**

Reserved, RES0.

**D, bit [9]**

Process state D mask. The possible values of this bit are:

<b>D</b>	<b>Meaning</b>
0	Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are not masked.
1	Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are masked.

When the target Exception level of the debug exception is higher than the current Exception level, the exception is not masked by this bit.

**A, bit [8]**

Error interrupt mask bit. The possible values of this bit are:

<b>A</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**I, bit [7]**

IRQ mask bit. The possible values of this bit are:

<b>I</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**F, bit [6]**

FIQ mask bit. The possible values of this bit are:

<b>F</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**Bit [5]**

Reserved, RES0.

**M[4], bit [4]**

Execution state that Debug state was entered from. Possible values of this bit are:

<b>M[4]</b>	<b>Meaning</b>
0	Exception taken from AArch64.

**M[3:0], bits [3:0]**

AArch64 state (Exception level and selected SP) that Debug state was entered from. The possible values are:

M[3:0]	State
0b0000	EL0t
0b0100	EL1t
0b0101	EL1h
0b1000	EL2t
0b1001	EL2h
0b1100	EL3t
0b1101	EL3h

Other values are reserved, and returning to an Exception level that is using AArch64 with a reserved value in this field is treated as an illegal exception return.

The bits in this field are interpreted as follows:

- M[3:2] holds the Exception Level.
- M[1] is unused and is RES0 for all non-reserved values.
- M[0] is used to select the SP:
  - 0 means the SP is always SP0.
  - 1 means the exception SP is determined by the EL.

## Accessing the DPSR\_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
DPSR_EL0	11	011	0100	0101	000

## 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.

Access to this register is from Debug state only. During normal execution this register is unallocated.

# ELR\_EL1, Exception Link Register (EL1)

The ELR\_EL1 characteristics are:

## Purpose

When taking an exception to EL1, holds the address to return to.

This register is part of the Special-purpose registers functional group.

## Configuration

There are no configuration notes.

## Attributes

ELR\_EL1 is a 64-bit register.

## Field descriptions

The ELR\_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
Return address																															
Return address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Return address.

An exception return from EL1 using AArch64 makes ELR\_EL1 become UNKNOWN.

## Accessing the ELR\_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
ELR_EL1	11	000	0100	0000	001
ELR_EL12	11	101	0100	0000	001

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
ELR_EL1	x	x	0	-	RW	n/a	RW

# ELR\_EL1, Exception Link Register (EL1)

ELR_EL1	0	0	1	-	RW	RW	RW
ELR_EL1	0	1	1	-	n/a	RW	RW
ELR_EL1	1	0	1	-	RW	<a href="#">ELR_EL2</a>	RW
ELR_EL1	1	1	1	-	n/a	<a href="#">ELR_EL2</a>	RW
ELR_EL12	x	x	0	-	-	n/a	-
ELR_EL12	0	0	1	-	-	-	-
ELR_EL12	0	1	1	-	n/a	-	-
ELR_EL12	1	0	1	-	-	RW	RW
ELR_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic ELR\_EL1 or ELR\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ELR\_EL2, Exception Link Register (EL2)

The ELR\_EL2 characteristics are:

## Purpose

When taking an exception to EL2, holds the address to return to.

This register is part of the Special-purpose registers functional group.

## Configuration

AArch64 System register ELR\_EL2 is architecturally mapped to AArch32 System register [ELR\\_hyp](#).

## Attributes

ELR\_EL2 is a 64-bit register.

## Field descriptions

The ELR\_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
Return address																															
Return address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Return address.

An exception return from EL2 using AArch64 makes ELR\_EL2 become UNKNOWN.

When EL2 is in AArch32 Execution state and an exception is taken from EL0, EL1, or EL2 to EL3 and AArch64 execution, the upper 32-bits of ELR\_EL2 are either set to 0 or hold the same value that they did before AArch32 execution. Which option is adopted is determined by an implementation, and might vary dynamically within an implementation. Correspondingly software must regard the value as being an UNKNOWN choice between the two values.

## Accessing the ELR\_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
ELR_EL2	11	100	0100	0000	001
ELR_EL1	11	000	0100	0000	001

## Accessibility

The register is accessible as follows:



<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
ELR_EL2	x	x	0	-	-	n/a	RW
ELR_EL2	0	0	1	-	-	RW	RW
ELR_EL2	0	1	1	-	n/a	RW	RW
ELR_EL2	1	0	1	-	-	RW	RW
ELR_EL2	1	1	1	-	n/a	RW	RW
ELR_EL1	x	x	0	-	<a href="#">ELR_EL1</a>	n/a	<a href="#">ELR_EL1</a>
ELR_EL1	0	0	1	-	<a href="#">ELR_EL1</a>	<a href="#">ELR_EL1</a>	<a href="#">ELR_EL1</a>
ELR_EL1	0	1	1	-	n/a	<a href="#">ELR_EL1</a>	<a href="#">ELR_EL1</a>
ELR_EL1	1	0	1	-	<a href="#">ELR_EL1</a>	RW	<a href="#">ELR_EL1</a>
ELR_EL1	1	1	1	-	n/a	RW	<a href="#">ELR_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic ELR\_EL2 or ELR\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ELR\_EL3, Exception Link Register (EL3)

The ELR\_EL3 characteristics are:

## Purpose

When taking an exception to EL3, holds the address to return to.

This register is part of the Special-purpose registers functional group.

## Configuration

There are no configuration notes.

## Attributes

ELR\_EL3 is a 64-bit register.

## Field descriptions

The ELR\_EL3 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
Return address																															
Return address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Return address.

An exception return from EL3 using AArch64 makes ELR\_EL3 become UNKNOWN.

## Accessing the ELR\_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
ELR_EL3	11	110	0100	0000	001

## 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

## ELR\_EL3, Exception Link Register (EL3)

x	1	1	-	n/a	-	RW
---	---	---	---	-----	---	----

This table applies to all instructions that can access this register.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ESR\_EL1, Exception Syndrome Register (EL1)

The ESR\_EL1 characteristics are:

## Purpose

Holds syndrome information for an exception taken to EL1.

This register is part of the Exception and fault handling registers functional group.

## Configuration

AArch64 System register ESR\_EL1 is architecturally mapped to AArch32 System register [DFSR](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

ESR\_EL1 is a 32-bit register.

## Field descriptions

See [ESR\\_ELx](#).

## Accessing the ESR\_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
ESR_EL1	11	000	0101	0010	000
ESR_EL12	11	101	0101	0010	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
ESR_EL1	x	x	0	-	RW	n/a	RW
ESR_EL1	0	0	1	-	RW	RW	RW
ESR_EL1	0	1	1	-	n/a	RW	RW
ESR_EL1	1	0	1	-	RW	<a href="#">ESR_EL2</a>	RW
ESR_EL1	1	1	1	-	n/a	<a href="#">ESR_EL2</a>	RW
ESR_EL12	x	x	0	-	-	n/a	-
ESR_EL12	0	0	1	-	-	-	-
ESR_EL12	0	1	1	-	n/a	-	-

ESR_EL12	1	0	1	-	-	RW	RW
ESR_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic ESR\_EL1 or ESR\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

## 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.

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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ESR\_EL2, Exception Syndrome Register (EL2)

The ESR\_EL2 characteristics are:

## Purpose

Holds syndrome information for an exception taken to EL2.

This register is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.

## Configuration

AArch64 System register ESR\_EL2 is architecturally mapped to AArch32 System register [HSR](#).

If EL2 is not implemented, this register is RES0 from EL3.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

ESR\_EL2 is a 32-bit register.

## Field descriptions

See [ESR\\_ELx](#).

## Accessing the ESR\_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
ESR_EL2	11	100	0101	0010	000
ESR_EL1	11	000	0101	0010	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
ESR_EL2	x	x	0	-	-	n/a	RW
ESR_EL2	0	0	1	-	-	RW	RW
ESR_EL2	0	1	1	-	n/a	RW	RW
ESR_EL2	1	0	1	-	-	RW	RW
ESR_EL2	1	1	1	-	n/a	RW	RW

# ESR\_EL2, Exception Syndrome Register (EL2)

ESR_EL1	x	x	0	-	<a href="#">ESR_EL1</a>	n/a	<a href="#">ESR_EL1</a>
ESR_EL1	0	0	1	-	<a href="#">ESR_EL1</a>	<a href="#">ESR_EL1</a>	<a href="#">ESR_EL1</a>
ESR_EL1	0	1	1	-	n/a	<a href="#">ESR_EL1</a>	<a href="#">ESR_EL1</a>
ESR_EL1	1	0	1	-	<a href="#">ESR_EL1</a>	RW	<a href="#">ESR_EL1</a>
ESR_EL1	1	1	1	-	n/a	RW	<a href="#">ESR_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic ESR\_EL2 or ESR\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ESR\_EL3, Exception Syndrome Register (EL3)

The ESR\_EL3 characteristics are:

## Purpose

Holds syndrome information for an exception taken to EL3.

This register is part of the Exception and fault handling registers functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

ESR\_EL3 is a 32-bit register.

## Field descriptions

See [ESR\\_ELx](#).

## Accessing the ESR\_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
ESR_EL3	11	110	0101	0010	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
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.





# ESR\_ELx, Exception Syndrome Register (ELx)

This describes [ESR\\_EL1](#), [ESR\\_EL2](#), and [ESR\\_EL3](#).

The ESR\_ELx characteristics are:

## Purpose

Holds syndrome information for an exception taken to ELx.

This register is part of the Exception and fault handling registers functional group.

## Usage constraints

## Traps and Enables

There are no traps or enables affecting this register.

## Configuration

If EL2 is not implemented, [ESR\\_EL2](#) is RES0 from EL3.

## Attributes

The ESR\_ELx registers are 32-bit registers.

## Field descriptions

The ESR\_ELx 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
EC						IL	ISS																								

ESR\_ELx is made UNKNOWN as a result of an exception return from ELx.

When an UNPREDICTABLE instruction is treated as UNDEFINED, and the exception is taken to ELx, the value of ESR\_ELx is UNKNOWN. The value written to ESR\_ELx must be consistent with a value that could be created as a result of an exception from the same Exception level that generated the exception as a result of a situation that is not UNPREDICTABLE at that Exception level, in order to avoid the possibility of a privilege violation.

### EC, bits [31:26]

Exception Class. Indicates the reason for the exception that this register holds information about.

For each EC value, the table references a subsection that gives information about:

- The cause of the exception, for example the configuration required to enable the trap.
- The encoding of the associated ISS.

Possible values of the EC field are:

EC	Meaning	ISS	Applies to
000000	Unknown reason.	<a href="#">Exceptions with an unknown reason</a>	All
000001	Trapped WFI or WFE instruction execution. Conditional WFE and WFI instructions that fail their condition code check do not cause an exception.	<a href="#">Exception from a WFI or WFE instruction</a>	All
000011	Trapped MCR or MRC access with (coproc==1111) that is not reported using EC 0b000000.	<a href="#">Exception from an MCR or MRC access</a>	All
000100	Trapped MCRR or MRRC access with (coproc==1111) that is not reported using EC 0b000000.	<a href="#">Exception from an MCRR or MRRC access</a>	All
000101	Trapped MCR or MRC access with (coproc==1110).	<a href="#">Exception from an MCR or MRC access</a>	All
000110	Trapped LDC or STC access. The only architected uses of these instructions are: <ul style="list-style-type: none"> <li>An STC to write data to memory from <a href="#">DBGDTRRXint</a>.</li> <li>An LDC to read data from memory to <a href="#">DBGDTRTXint</a>.</li> </ul>	<a href="#">Exception from an LDC or STC instruction</a>	All
000111	Access to SVE, Advanced SIMD, or floating-point functionality trapped by <a href="#">CPACR_EL1.FPEN</a> , <a href="#">CPTR_EL2.FPEN</a> , <a href="#">CPTR_EL2.TFP</a> , or <a href="#">CPTR_EL3.TFP</a> control. Excludes exceptions resulting from <a href="#">CPACR_EL1</a> when the value of <a href="#">HCR_EL2.TGE</a> is 1, or because SVE or Advanced SIMD and floating-point are not implemented. These are reported with EC value 0b000000 as described in 'EC encodings when routing exceptions to EL2' in the ARMv8 ARM, section D1.10.4.	<a href="#">Exception from an access to an Advanced SIMD or floating-point register, resulting from <a href="#">CPACR_EL1.FPEN</a> or <a href="#">CPTR_ELx.TFP</a></a>	All
001000	Trapped VMRS access, from ID group trap, that is not reported using EC 0b000111.	<a href="#">Exception from an MCR or MRC access</a>	<a href="#">ESR_EL2</a>
001100	Trapped MRRC access with (coproc==1110).	<a href="#">Exception from an MCRR or MRRC access</a>	All
001110	Illegal Execution state.	<a href="#">Exception from an Illegal Execution state, or a PC or SP alignment fault</a>	All
010001	SVC instruction execution in AArch32 state. This is reported in ESR_EL2 only when the exception is generated because the value of <a href="#">HCR_EL2.TGE</a> is 1.	<a href="#">Exception from HVC or SVC instruction execution</a>	<a href="#">ESR_EL1</a> and <a href="#">ESR_EL2</a>
010010	HVC instruction execution in AArch32 state, when HVC is not disabled.	<a href="#">Exception from HVC or SVC instruction execution</a>	<a href="#">ESR_EL2</a>
010011	SMC instruction execution in AArch32 state, when SMC is not disabled. This is reported in ESR_EL2 only when the exception is generated because the value of <a href="#">HCR_EL2.TSC</a> is 1.	<a href="#">Exception from SMC instruction execution in AArch32 state</a>	<a href="#">ESR_EL2</a> and <a href="#">ESR_EL3</a>
010101	SVC instruction execution in AArch64 state.	<a href="#">Exception from HVC or SVC instruction execution</a>	All
010110	HVC instruction execution in AArch64 state, when HVC is not disabled.	<a href="#">Exception from HVC or SVC instruction execution</a>	<a href="#">ESR_EL2</a> and <a href="#">ESR_EL3</a>

010111	SMC instruction execution in AArch64 state, when SMC is not disabled. This is reported in ESR_EL2 only when the exception is generated because the value of <a href="#">HCR_EL2.TSC</a> is 1.	<a href="#">Exception from SMC instruction execution in AArch64 state</a>	<a href="#">ESR_EL2</a> and <a href="#">ESR_EL3</a>
011000	Trapped MSR, MRS or System instruction execution in AArch64 state, that is not reported using EC 0b000000, 0b000001 or 0b000111. This include all instructions that cause exceptions that are part of the encoding space defined in 'System instruction class encoding overview' in the ARMv8 ARM, section C5.2.2, except for those exceptions reported using EC values 0b000000, 0b000001, or 0b000111.	<a href="#">Exception from MSR, MRS, or System instruction execution in AArch64 state</a>	All
011001	Access to SVE functionality as a result of <a href="#">CPACR_EL1.ZEN</a> , <a href="#">CPTR_EL2.ZEN</a> , <a href="#">CPTR_EL2.TZ</a> , or <a href="#">CPTR_EL3.EZ</a> , except those reported using EC value 0b000000. This EC is defined only if SVE is implemented.	<a href="#">Exception from an access to SVE functionality, resulting from CPACR_EL1.ZEN, CPTER_EL2.ZEN, CPTER_EL2.TZ, or CPTER_EL3.EZ</a>	All
011111	IMPLEMENTATION DEFINED exception to EL3.	<a href="#">IMPLEMENTATION DEFINED exception to EL3</a>	<a href="#">ESR_EL3</a>
100000	Instruction Abort from a lower Exception level. Used for MMU faults generated by instruction accesses and Synchronous external aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.	<a href="#">Exception from an Instruction Abort</a>	All
100001	Instruction Abort taken without a change in Exception level. Used for MMU faults generated by instruction accesses and Synchronous external aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.	<a href="#">Exception from an Instruction Abort</a>	All
100010	PC alignment fault exception.	<a href="#">Exception from an Illegal Execution state, or a PC or SP alignment fault</a>	All
100100	Data Abort from a lower Exception level. Used for MMU faults generated by data accesses, alignment faults other than those caused by the Stack Pointer misalignment, and Synchronous external aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.	<a href="#">Exception from a Data Abort</a>	All
100101	Data Abort taken without a change in Exception level. Used for MMU faults generated by data accesses, alignment faults other than those caused by the Stack Pointer misalignment, and Synchronous external aborts, including synchronous parity or ECC errors. Not used for debug related exceptions.	<a href="#">Exception from a Data Abort</a>	All
100110	SP alignment fault exception.	<a href="#">Exception from an Illegal Execution state, or a PC or SP alignment fault</a>	All

101000	Trapped floating-point exception taken from AArch32 state. Whether this Exception class is supported is IMPLEMENTATION DEFINED.	<a href="#">Exception from a trapped floating-point exception</a>	<a href="#">ESR_EL1</a> and <a href="#">ESR_EL2</a>
101100	Trapped floating-point exception taken from AArch64 state. Whether this Exception class is supported is IMPLEMENTATION DEFINED.	<a href="#">Exception from a trapped floating-point exception</a>	All
101111	SError interrupt.	<a href="#">SError interrupt</a>	All
110000	Breakpoint exception from a lower Exception level.	<a href="#">Exception from a Breakpoint or Vector Catch debug exception</a>	<a href="#">ESR_EL1</a> and <a href="#">ESR_EL2</a>
110001	Breakpoint exception taken without a change in Exception level.	<a href="#">Exception from a Breakpoint or Vector Catch debug exception</a>	<a href="#">ESR_EL1</a> and <a href="#">ESR_EL2</a>
110010	Software Step exception from a lower Exception level.	<a href="#">Exception from a Software Step exception</a>	<a href="#">ESR_EL1</a> and <a href="#">ESR_EL2</a>
110011	Software Step exception taken without a change in Exception level.	<a href="#">Exception from a Software Step exception</a>	<a href="#">ESR_EL1</a> and <a href="#">ESR_EL2</a>
110100	Watchpoint exception from a lower Exception level.	<a href="#">Exception from a Watchpoint exception</a>	<a href="#">ESR_EL1</a> and <a href="#">ESR_EL2</a>
110101	Watchpoint exception taken without a change in Exception level.	<a href="#">Exception from a Watchpoint exception</a>	<a href="#">ESR_EL1</a> and <a href="#">ESR_EL2</a>
111000	BKPT instruction execution in AArch32 state.	<a href="#">Exception from execution of a Breakpoint instruction</a>	<a href="#">ESR_EL1</a> and <a href="#">ESR_EL2</a>
111010	Vector Catch exception from AArch32 state. The only case where a Vector Catch exception is taken to an Exception level that is using AArch64 is when the exception is routed to EL2 and EL2 is using AArch64.	<a href="#">Exception from a Breakpoint or Vector Catch debug exception</a>	<a href="#">ESR_EL2</a>
111100	BRK instruction execution in AArch64 state. This is reported in ESR_EL3 only if a BRK instruction is executed at EL3.	<a href="#">Exception from execution of a Breakpoint instruction</a>	All

All other EC values are reserved by ARM, and:

- Unused values in the range 0b000000 - 0b101100 (0x00 - 0x2C) are reserved for future use for synchronous exceptions.
- Unused values in the range 0b101101 - 0b111111 (0x2D - 0x3F) are reserved for future use, and might be used for synchronous or asynchronous exceptions.

The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in System and memory-mapped registers and translation table entries' in the ARM ARM, section K1.1.11.

## IL, bit [25]

Instruction Length for synchronous exceptions. Possible values of this bit are:

IL	Meaning
0	16-bit instruction trapped.
1	32-bit instruction trapped. This value is also used when the exception is one of the following: <ul style="list-style-type: none"> <li>An SError interrupt.</li> <li>An Instruction Abort exception.</li> <li>A PC alignment fault exception.</li> <li>An SP alignment fault exception.</li> <li>A Data Abort exception for which the value of the ISV bit is 0.</li> <li>An Illegal Execution state exception.</li> <li>Any debug exception except for Breakpoint instruction exceptions. For Breakpoint instruction exceptions, this bit has its standard meaning: <div> <div>0</div> <div>16-bit T32 BKPT instruction.</div> <div>1</div> <div>32-bit A32 BKPT instruction or A64 BRK instruction.</div> </div> </li> </ul>
	<ul style="list-style-type: none"> <li>An exception reported using EC value 0b000000.</li> </ul>

## ISS, bits [24:0]

Instruction Specific Syndrome. Architecturally, this field can be defined independently for each defined Exception class. However, in practice, some ISS encodings are used for more than one Exception class.

Typically, an ISS encoding has a number of subfields. When an ISS subfield holds a register number, the value returned in that field is the AArch64 view of the register number. For an exception taken from AArch32 state, 'Mapping of the general-purpose registers between the Execution states' in the ARMv8 ARM, section D1.20.1, defines this view of the specified AArch32 register. If the AArch32 register descriptor is 0b1111, then:

- If the instruction that generated the exception was not UNPREDICTABLE, the field takes the value 0b11111.
- If the instruction that generated the exception was UNPREDICTABLE, the field takes an UNKNOWN value that must be either:
  - The AArch64 view of the register number of a register that might have been used at the Exception level from which the exception was taken.
  - The value 0b11111.

When the EC field is 0b000000, indicating an exception with an unknown reason, the ISS field is not valid, RES0.

## Exceptions with an unknown reason

This is the layout of the ISS field for exceptions with the following EC values:

- 0b000000, Unknown reason.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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

## Bits [24:0]

Reserved, RES0.

This EC code is used for all exceptions that are not covered by any other EC value. This includes exceptions that are generated in the following situations:

- The attempted execution of an instruction bit pattern that has no allocated instruction at the current Exception level and Security state, including:
  - A read access using a System register pattern that is not allocated for reads at the current Exception level and Security state.
  - A write access using a System register pattern that is not allocated for writes at the current Exception level and Security state.
  - Instruction encodings for instructions not implemented in the implementation.
- In Debug state, the attempted execution of an instruction bit pattern that is unallocated in Debug state.
- In Non-debug state, the attempted execution of an instruction bit pattern that is unallocated in Non-debug state.
- In AArch32 state, attempted execution of a short vector floating-point instruction.
- In an implementation that does not include Advanced SIMD and floating-point functionality, an attempted access to Advanced SIMD or floating-point functionality under conditions where that access would be permitted if that functionality was present. This includes the attempted execution of an Advanced SIMD or floating-point instruction, and attempted accesses to Advanced SIMD and floating-point System registers.

- An exception generated because of the value of one of the [SCTLR\\_EL1](#).{ITD, SED, CP15BEN} control bits.
- Attempted execution of:
  - An HVC instruction when disabled by [HCR\\_EL2](#).HCD or [SCR\\_EL3](#).HCE.
  - An SMC instruction when disabled by [SCR\\_EL3](#).SMD.
  - An HLT instruction when disabled by [EDSCR](#).HDE.
- Attempted execution of an MSR or MRS instruction to access [SP\\_EL0](#) when the value of [SPSel.SP](#) is 0.
- Attempted execution, in Debug state, of:
  - A DCPS1 instruction in Non-secure state from EL0 when the value of [HCR\\_EL2](#).TGE is 1.
  - A DCPS2 instruction from EL1 or EL0 when the value of [SCR\\_EL3](#).NS is 0, or when EL2 is not implemented.
  - A DCPS3 instruction when the value of [EDSCR](#).SDD is 1, or when EL3 is not implemented.
- When EL3 is using AArch64, attempted execution from Secure EL1 of an SRS instruction using R13\_mon. See 'Traps to EL3 of monitor functionality from Secure EL1 using AArch32' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).
- In Debug state when the value of [EDSCR](#).SDD is 1, the attempted execution at EL2, EL1, or EL0 of an instruction that is configured to trap to EL3.
- In AArch32 state, the attempted execution of an MRS (Banked register) or an MSR (Banked register) instruction to [SPSR\\_mon](#), [SP\\_mon](#), or [LR\\_mon](#).
- An exception that is taken to EL2 because the value of [HCR\\_EL2](#).TGE is 1 that, if the value of [HCR\\_EL2](#).TGE was 0 would have been reported with an ESR\_ELx.EC value of 0b000111.
- In an implementation that does not include SVE, execution of an SVE instruction or an instruction that accesses the [ID\\_AA64ZFR0\\_EL1](#), [ZCR\\_EL1](#), [ZCR\\_EL2](#), and [ZCR\\_EL3](#) registers.

## Exception from a WFI or WFE instruction

This is the layout of the ISS field for exceptions with the following EC values:

- 0b000001, Trapped WFI or WFE instruction execution.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TI

### CV, bit [24]

Condition code valid. Possible values of this bit are:

CV	Meaning
0	The COND field is not valid.
1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

### COND, bits [23:20]

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch64, this field is set to 0b1110.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.

- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

**Bits [19:1]**

Reserved, RES0.

**TI, bit [0]**

Trapped instruction. Possible values of this bit are:

TI	Meaning
0	WFI trapped.
1	WFE trapped.

The following sections describe configuration settings for generating this exception:

- 'Controls for exceptions taken to EL1 using AArch64' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).
- 'Traps to EL2 of Non-secure EL1 and EL0 execution of WFE and WFI instructions' in the ARMv8 ARM, section D1.
- 'Traps to EL3 of EL2, EL1, and EL0 execution of WFE and WFI instructions' in the ARMv8 ARM, section D1.

**Exception from an MCR or MRC access**

This is the layout of the ISS field for exceptions with the following EC values:

- 0b000011, Trapped MCR or MRC access with (coproc==1111) that is not reported using EC 0b000000.
- 0b000101, Trapped MCR or MRC access with (coproc==1110).
- 0b001000, Trapped VMRS access, from ID group trap, that is not reported using EC 0b000111.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				Opc2		Opc1		CRn				Rt				CRm				Direction			

**CV, bit [24]**

Condition code valid. Possible values of this bit are:

CV	Meaning
0	The COND field is not valid.
1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

**COND, bits [23:20]**

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch64, this field is set to 0b1110.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:



- CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
- CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

**Opc2, bits [19:17]**

The Opc2 value from the issued instruction.

For a trapped VMRS access, holds the value 0b000.

**Opc1, bits [16:14]**

The Opc1 value from the issued instruction.

For a trapped VMRS access, holds the value 0b111.

**CRn, bits [13:10]**

The CRn value from the issued instruction.

For a trapped VMRS access, holds the reg field from the VMRS instruction encoding.

**Rt, bits [9:5]**

The Rt value from the issued instruction, the general-purpose register used for the transfer. The reported value gives the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states' in the ARMv8 ARM, section D1.20.1.

**CRm, bits [4:1]**

The CRm value from the issued instruction.

For a trapped VMRS access, holds the value 0b0000.

**Direction, bit [0]**

Indicates the direction of the trapped instruction. The possible values of this bit are:

Direction	Meaning
0	Write to System register space. MCR instruction.
1	Read from System register space. MRC or VMRS instruction.

The following sections describe configuration settings for generating exceptions that are reported using EC value 0b000011:

- 'Traps to EL1 of EL0 accesses to the Generic Timer registers' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).
- 'Traps to EL1 of EL0 accesses to Performance Monitors registers' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).
- 'Traps to EL2 of Non-secure EL1 accesses to virtual memory control registers' in the ARMv8 ARM, section D1.
- 'Traps to EL2 of Non-secure EL1 execution of TLB maintenance instructions' in the ARMv8 ARM, section D1.
- 'Traps to EL2 of Non-secure EL1 and EL0 execution of cache maintenance instructions' in the ARMv8 ARM, section D1.
- 'Traps to EL2 of Non-secure EL1 and EL0 accesses to the Auxiliary Control Register' in the ARMv8 ARM, section D1.
- 'Traps to EL2 of Non-secure EL1 and EL0 accesses to lockdown, DMA, and TCM operations' in the ARMv8 ARM, section D1.
- 'Traps to EL2 of Non-secure EL1 and EL0 accesses to the ID registers' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).
- 'Trapping to EL2 of Non-secure EL1 accesses to the CPACR\_EL1 or CPACR' in the ARMv8 ARM, section D1.
- 'Generic trapping to EL2 of Non-secure EL1 and EL0 accesses to System registers, from AArch32 state only' in the ARMv8 ARM, section D1.
- 'Traps to EL2 of Non-secure EL0 and EL1 accesses to the Generic Timer registers' in the ARMv8 ARM, section D1.
- 'Traps to EL2 of Non-secure EL1 and EL0 accesses to Performance Monitors registers' in the ARMv8 ARM, section D1.

- 'Traps to EL3 of Secure monitor functionality from Secure EL1 using AArch32' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).
- 'Trapping to EL3 of EL2 accesses to the CPTR\_EL2 or HCPTR, and EL2 and EL1 accesses to the CPACR\_EL1 or CPACR' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).
- 'Traps to EL3 of EL2, EL1, and EL0 accesses to Performance Monitors registers' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).

The following sections describe configuration settings for generating exceptions that are reported using EC value 0b000101:

- 'Traps to EL1 of EL0 and EL1 System register accesses to the trace registers' in the ARMv8 ARM, section D1.
- 'Traps to EL1 of EL0 accesses to the Debug Communications Channel (DCC) registers' in the ARMv8 ARM, section D1.
- 'Traps to EL2 of Non-secure EL1 and EL0 accesses to the ID registers' in the ARMv8 ARM, section D1, for trapped accesses to the JIDR.
- 'Traps to EL2 of Non-secure system register accesses to the trace registers' in the ARMv8 ARM, section D1.
- 'Trapping System register accesses to Debug ROM registers to EL2' in the ARMv8 ARM, section D1.
- 'Trapping System register accesses to OS-related debug registers to EL2' in the ARMv8 ARM, section D1.
- 'Trapping general System register accesses to debug registers to EL2' in the ARMv8 ARM, section D1.
- 'Traps to EL3 of all System register accesses to the trace registers' in the ARMv8 ARM, section D1.
- 'Trapping System register accesses to OS-related debug registers to EL3' in the ARMv8 ARM, section D1.
- 'Trapping general System register accesses to debug registers to EL3' in the ARMv8 ARM, section D1.

'Traps to EL2 of Non-secure EL1 and EL0 accesses to the ID registers' in the ARMv8 ARM, section D1, describes configuration settings for generating exceptions that are reported using EC value 0b001000.

## Exception from an MCRR or MRRC access

This is the layout of the ISS field for exceptions with the following EC values:

- 0b000100, Trapped MCRR or MRRC access with (coproc==1111) that is not reported using EC 0b000000.
- 0b001100, Trapped MRRC access with (coproc==1110).

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				Opc1				0	Rt2				Rt				CRm				Direction		

### CV, bit [24]

Condition code valid. Possible values of this bit are:

CV	Meaning
0	The COND field is not valid.
1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

### COND, bits [23:20]

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch64, this field is set to 0b1110.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.

- CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

**Opc1, bits [19:16]**

The Opc1 value from the issued instruction.

**Bit [15]**

Reserved, RES0.

**Rt2, bits [14:10]**

The Rt2 value from the issued instruction, the second general-purpose register used for the transfer. The reported value gives the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states' in the ARMv8 ARM, section D1.20.1.

**Rt, bits [9:5]**

The Rt value from the issued instruction, the first general-purpose register used for the transfer. The reported value gives the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states' in the ARMv8 ARM, section D1.20.1.

**CRm, bits [4:1]**

The CRm value from the issued instruction.

**Direction, bit [0]**

Indicates the direction of the trapped instruction. The possible values of this bit are:

Direction	Meaning
0	Write to System register space. MCRR instruction.
1	Read from System register space. MRRC instruction.

The following sections describe configuration settings for generating exceptions that are reported using EC value 0b000100:

- 'Traps to EL1 of EL0 accesses to the Generic Timer registers' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).
- 'Traps to EL1 of EL0 accesses to Performance Monitors registers' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).
- 'Traps to EL2 of Non-secure EL1 accesses to virtual memory control registers' in the ARMv8 ARM, section D1.
- 'General trapping to EL2 of Non-secure EL0 and EL1 accesses to System registers, from AArch32 state only' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).
- 'Traps to EL2 of Non-secure EL0 and EL1 accesses to the Generic Timer registers' in the ARMv8 ARM, section D1.
- 'Traps to EL2 of Non-secure EL0 and EL1 accesses to Performance Monitors registers' in the ARMv8 ARM, section D1.
- 'Traps to EL3 of EL2, EL1, and EL0 accesses to Performance Monitors registers' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).

The following sections describe configuration settings for generating exceptions that are reported using EC value 0b001100:

- 'Traps to EL1 of EL0 and EL1 System register accesses to the trace registers' in the ARMv8 ARM, section D1.
- 'Traps to EL1 of EL0 accesses to the Debug Communications Channel (DCC) registers' in the ARMv8 ARM, section D1.
- 'Traps to EL2 of Non-secure system register accesses to the trace registers' in the ARMv8 ARM, section D1.
- 'Trapping System register accesses to Debug ROM registers' in the ARMv8 ARM, section D1.
- 'Traps to EL3 of all System register accesses to the trace registers' in the ARMv8 ARM, section D1.
- 'Trapping System register accesses to OS-related debug registers' in the ARMv8 ARM, section D1.

**Exception from an LDC or STC instruction**

This is the layout of the ISS field for exceptions with the following EC values:

- 0b000110, Trapped LDC or STC access.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV	COND				imm8								0	0	Rn				Offset	AM		Direction		

**CV, bit [24]**

Condition code valid. Possible values of this bit are:

CV	Meaning
0	The COND field is not valid.
1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

**COND, bits [23:20]**

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch64, this field is set to 0b1110.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

**imm8, bits [19:12]**

The immediate value from the issued instruction.

**Bits [11:10]**

Reserved, RES0.

**Rn, bits [9:5]**

The Rn value from the issued instruction, the general-purpose register used for the transfer. The reported value gives the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states' in the ARMv8 ARM, section D1.20.1.

This field is valid only when AM[2] is 0, indicating an immediate form of the LDC or STC instruction. When AM[2] is 1, indicating a literal form of the LDC or STC instruction, this field is UNKNOWN.

**Offset, bit [4]**

Indicates whether the offset is added or subtracted:

Offset	Meaning
0	Subtract offset.
1	Add offset.

This bit corresponds to the U bit in the instruction encoding.

### AM, bits [3:1]

Addressing mode. The permitted values of this field are:

AM	Meaning
000	Immediate unindexed.
001	Immediate post-indexed.
010	Immediate offset.
011	Immediate pre-indexed.
100	Literal unindexed. LDC instruction in A32 instruction set only. For a trapped STC instruction or a trapped T32 LDC instruction this encoding is reserved.
110	Literal offset. LDC instruction only. For a trapped STC instruction, this encoding is reserved.

The values 0b101 and 0b111 are reserved. The effect of programming this field to a reserved value is that behavior is CONSTRAINED UNPREDICTABLE, as described in 'Reserved values in AArch64 System registers and translation table entries' in the ARM ARM, section K1.2.2.

Bit [2] in this subfield indicates the instruction form, immediate or literal.

Bits [1:0] in this subfield correspond to the bits {P, W} in the instruction encoding.

### Direction, bit [0]

Indicates the direction of the trapped instruction. The possible values of this bit are:

Direction	Meaning
0	Write to memory. STC instruction.
1	Read from memory. LDC instruction.

The following sections describe the configuration settings for the traps that are reported using EC value 0b000110:

- 'Traps to EL1 of EL0 accesses to the Debug Communications Channel (DCC) registers' in the ARMv8 ARM, section D1.
- 'Trapping general System register accesses to debug registers to EL2' in the ARMv8 ARM, section D1.
- 'Trapping general System register accesses to debug registers to EL3' in the ARMv8 ARM, section D1.

## Exception from an access to an Advanced SIMD or floating-point register, resulting from CPACR\_EL1.FPEN or CPTR\_ELx.TFP

This is the layout of the ISS field for exceptions with the following EC values:

- 0b000111, Access to SVE, Advanced SIMD, or floating-point functionality trapped by CPACR\_EL1.FPEN, CPTR\_EL2.FPEN, CPTR\_EL2.TFP, or CPTR\_EL3.TFP control.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV		COND			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

The accesses covered by this trap include:

- Execution of SVE or Advanced SIMD and floating-point instructions.
- Accesses to the Advanced SIMD and floating-point System registers.

For an implementation that does not include SVE, floating-point or Advanced SIMD, the exception is reported using the EC value 0b000000.

**CV, bit [24]**

Condition code valid. Possible values of this bit are:

CV	Meaning
0	The COND field is not valid.
1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

**COND, bits [23:20]**

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch64, this field is set to 0b1110.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

**Bits [19:0]**

Reserved, RES0.

The following sections describe the configuration settings for the traps that are reported using EC value 0b000111:

- 'Traps to EL1 of EL0 and EL1 accesses to SIMD and floating-point functionality' in the ARMv8 ARM, section D1.
- 'General trapping to EL2 of Non-secure accesses to the SIMD and floating-point registers' in the ARMv8 ARM, section D1.
- 'Traps to EL3 of all System register accesses to the trace registers' in the ARMv8 ARM, section D1.

## Exception from an access to SVE functionality, resulting from CPACR\_EL1.ZEN, CPTR\_EL2.ZEN, CPTR\_EL2.TZ, or CPTR\_EL3.EZ

This is the layout of the ISS field for exceptions with the following EC values:

- 0b011001, Access to SVE functionality as a result of CPACR\_EL1.ZEN, CPTR\_EL2.ZEN, CPTR\_EL2.TZ, or CPTR\_EL3.EZ, except those reported using EC value 0b000000.

24	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

**Bits [24:0]**

Reserved, RES0.

The accesses covered by this trap include:

- Execution of SVE instructions.
- Accesses to the SVE system registers.

When an implementation does not include SVE, the exception is reported using the EC value 0b000000.

## Exception from an Illegal Execution state, or a PC or SP alignment fault

This is the layout of the ISS field for exceptions with the following EC values:

- 0b001110, Illegal Execution state.
- 0b100010, PC alignment fault exception.
- 0b100110, SP alignment fault exception.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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

### Bits [24:0]

Reserved, RES0.

There are no configuration settings for generating Illegal Execution state exceptions and PC alignment fault exceptions. 'Stack pointer alignment checking' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model), describes the configuration settings for generating SP alignment fault exceptions.

## Exception from HVC or SVC instruction execution

This is the layout of the ISS field for exceptions with the following EC values:

- 0b010001, SVC instruction execution in AArch32 state.
- 0b010010, HVC instruction execution in AArch32 state, when HVC is not disabled.
- 0b010101, SVC instruction execution in AArch64 state.
- 0b010110, HVC instruction execution in AArch64 state, when HVC is not disabled.

24	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																

imm16

### Bits [24:16]

Reserved, RES0.

### imm16, bits [15:0]

The value of the immediate field from the HVC or SVC instruction.

For an HVC instruction, and for an A64 SVC instruction, this is the value of the imm16 field of the issued instruction.

For an A32 or T32 SVC instruction:

- If the instruction is unconditional, then:
  - For the T32 instruction, this field is zero-extended from the imm8 field of the instruction.
  - For the A32 instruction, this field is the bottom 16 bits of the imm24 field of the instruction.
- If the instruction is conditional, this field is UNKNOWN.

In AArch32 state, the HVC instruction is unconditional, and a conditional SVC instruction generates an exception only if it passes its condition code check. Therefore, the syndrome information for these exceptions does not require conditionality information.

For T32 and A32 instructions, see 'SVC' in the ARMv8 ARM, section F7 (T32 and A32 Base Instruction Set Instruction Descriptions), and 'HVC' in the ARMv8 ARM, section F7.

For A64 instructions, see 'SVC' in the ARMv8 ARM, section C5 (A64 Base Instruction Descriptions), and 'HVC' in the ARMv8 ARM, section C5.

## Exception from SMC instruction execution in AArch32 state

This is the layout of the ISS field for exceptions with the following EC values:

- 0b010011, SMC instruction execution in AArch32 state, when SMC is not disabled.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CV		COND		CCKNOWNPASS		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

For an SMC instruction that completes normally and generates an exception that is taken to EL3, the ISS encoding is RES0.

For an SMC instruction that is trapped to EL2 from Non-secure EL1 because [HCR\\_EL2.TSC](#) is 1, the ISS encoding is as shown in the diagram.

### CV, bit [24]

Condition code valid. Possible values of this bit are:

CV	Meaning
0	The COND field is not valid.
1	The COND field is valid.

For exceptions taken from AArch64, CV is set to 1.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether CV is set to 1 or set to 0. See the description of the COND field for more information.

This field is only valid if CCKNOWNPASS is 1, otherwise it is RES0.

### COND, bits [23:20]

The condition code for the trapped instruction. This field is valid only for exceptions taken from AArch32, and only when the value of CV is 1.

For exceptions taken from AArch64, this field is set to 0b1110.

For exceptions taken from AArch32:

- When an A32 instruction is trapped, CV is set to 1 and:
  - If the instruction is conditional, COND is set to the condition code field value from the instruction.
  - If the instruction is unconditional, COND is set to 0b1110.
- A conditional A32 instruction that is known to pass its condition code check can be presented either:
  - With COND set to 0b1110, the value for unconditional.
  - With the COND value held in the instruction.
- When a T32 instruction is trapped, it is IMPLEMENTATION DEFINED whether:
  - CV is set to 0 and COND is set to an UNKNOWN value. Software must examine the SPSR.IT field to determine the condition, if any, of the T32 instruction.
  - CV is set to 1 and COND is set to the condition code for the condition that applied to the instruction.
- For an implementation that, for both A32 and T32 instructions, takes an exception on a trapped conditional instruction only if the instruction passes its condition code check, these definitions mean that when CV is set to 1 it is IMPLEMENTATION DEFINED whether the COND field is set to 0b1110, or to the value of any condition that applied to the instruction.

This field is only valid if CCKNOWNPASS is 1, otherwise it is RES0.

### CCKNOWNPASS, bit [19]

Indicates whether the instruction might have failed its condition code check.

CCKNOWNPASS	Meaning
0	The instruction was unconditional, or was conditional and passed its condition code check.
1	The instruction was conditional, and might have failed its condition code check.

#### Note

In an implementation in which an SMC instruction that fails its code check is not trapped, this field can always return the value 0.



**Bits [18:0]**

Reserved, RES0.

'Traps to EL2 of Non-secure EL1 execution of SMC instructions' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model), describes the configuration settings for trapping SMC instructions from Non-secure EL1 modes, and 'System calls' in the ARMv8 ARM, section D1.16, describes the case where these exceptions are trapped to EL3.

**Exception from SMC instruction execution in AArch64 state**

This is the layout of the ISS field for exceptions with the following EC values:

- 0b010111, SMC instruction execution in AArch64 state, when SMC is not disabled.

24	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																

imm16

**Bits [24:16]**

Reserved, RES0.

**imm16, bits [15:0]**

The value of the immediate field from the issued SMC instruction.

The value of ISS[24:0] described here is used both:

- When an SMC instruction is trapped from Non-secure EL1 modes.
- When an SMC instruction is not trapped, so completes normally and generates an exception that is taken to EL3.

'Traps to EL2 of Non-secure EL1 execution of SMC instructions' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model), describes the configuration settings for trapping SMC instructions from Non-secure EL1 modes, and 'System calls' in the ARMv8 ARM, section D1.16, describes the case where these exceptions are trapped to EL3.

**Exception from MSR, MRS, or System instruction execution in AArch64 state**

This is the layout of the ISS field for exceptions with the following EC values:

- 0b011000, Trapped MSR, MRS or System instruction execution in AArch64 state, that is not reported using EC 0b000000, 0b000001 or 0b000111.

24	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	Op0		Op2			Op1			CRn						Rt				CRm			Direction

**Bits [24:22]**

Reserved, RES0.

**Op0, bits [21:20]**

The Op0 value from the issued instruction.

**Op2, bits [19:17]**

The Op2 value from the issued instruction.

**Op1, bits [16:14]**

The Op1 value from the issued instruction.

**CRn, bits [13:10]**

The CRn value from the issued instruction.

**Rt, bits [9:5]**

The Rt value from the issued instruction, the general-purpose register used for the transfer.

**CRm, bits [4:1]**

The CRm value from the issued instruction.

**Direction, bit [0]**

Indicates the direction of the trapped instruction. The possible values of this bit are:

Direction	Meaning
0	Write access, including MSR instructions.
1	Read access, including MRS instructions.

For exceptions caused by System instructions, see the 'System' subsection of 'Branches, exception generating and system instructions' in the ARMv8 ARM, section C3 (A64 Instruction Set Encoding), for the encoding values returned by an instruction.

The following sections describe configuration settings for generating the exception that is reported using EC value 0b011000:

- In 'EL1 configurable controls' in the ARMv8 ARM, section D1.
  - 'Traps to EL1 of EL0 execution of cache maintenance instructions' in the ARMv8 ARM, section D1.
  - 'Traps to EL1 of EL0 accesses to the CTR\_EL0' in the ARMv8 ARM, section D1.
  - 'Traps to EL1 of EL0 execution of DC ZVA instructions' in the ARMv8 ARM, section D1.
  - 'Traps to EL1 of EL0 accesses to the PSTATE.{D, A, I, F} interrupt masks' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).
  - 'Traps to EL1 of EL0 and EL1 System register accesses to the trace registers' in the ARMv8 ARM, section D1.
  - 'Traps to EL1 of EL0 accesses to the Debug Communications Channel (DCC) registers' in the ARMv8 ARM, section D1.
  - 'Traps to EL1 of EL0 accesses to the Generic Timer registers' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).
  - 'Traps to EL1 of EL0 accesses to Performance Monitors registers' in the ARMv8 ARM, section D1.
- In 'EL2 configurable controls' in the ARMv8 ARM, section D1.
  - 'Traps to EL2 of Non-secure EL1 accesses to virtual memory control registers' in the ARMv8 ARM, section D1.
  - 'Traps to EL2 of Non-secure EL0 and EL1 execution of DC ZVA instructions' in the ARMv8 ARM, section D1.
  - 'Traps to EL2 of Non-secure EL1 execution of TLB maintenance instructions' in the ARMv8 ARM, section D1.
  - 'Traps to EL2 of Non-secure EL0 and EL1 execution of cache maintenance instructions' in the ARMv8 ARM, section D1.
  - 'Traps to EL2 of Non-secure EL1 accesses to the Auxiliary Control Register' in the ARMv8 ARM, section D1.
  - 'Traps to EL2 of Non-secure EL0 and EL1 accesses to lockdown, DMA, and TCM operations' in the ARMv8 ARM, section D1.
  - 'Traps to EL2 of Non-secure EL0 and EL1 accesses to the ID registers' in the ARMv8 ARM, section D1.
  - 'Trapping to EL2 of Non-secure EL1 accesses to the CPACR\_EL1 or CPACR' in the ARMv8 ARM, section D1.
  - 'Traps to EL2 of Non-secure system register accesses to the trace registers' in the ARMv8 ARM, section D1.
  - 'Trapping System register accesses to Debug ROM registers to EL2' in the ARMv8 ARM, section D1.
  - 'Trapping System register accesses to OS-related debug registers to EL2' in the ARMv8 ARM, section D1.
  - 'Traps to EL2 of Non-secure EL0 and EL1 accesses to the Generic Timer registers' in the ARMv8 ARM, section D1.
  - 'Trapping general System register accesses to debug registers to EL2' in the ARMv8 ARM, section D1.
  - 'Traps to EL2 of Non-secure EL0 and EL1 accesses to Performance Monitors registers' in the ARMv8 ARM, section D1.
- In 'EL3 configurable controls' in the ARMv8 ARM, section D1.
  - 'Traps to EL3 of Secure EL1 accesses to the Counter-timer Physical Secure timer registers' in the ARMv8 ARM, section D1.
  - 'Trapping to EL3 of EL2 accesses to the CPTR\_EL2 or HCPTR, and EL2 and EL1 accesses to the CPACR\_EL1 or CPACR' in the ARMv8 ARM, section D1.
  - 'Traps to EL3 of all System register accesses to the trace registers' in the ARMv8 ARM, section D1.
  - 'Trapping System register accesses to OS-related debug registers to EL3' in the ARMv8 ARM, section D1.
  - 'Trapping general System register accesses to debug registers to EL3' in the ARMv8 ARM, section D1.
  - 'Traps to EL3 of EL2, EL1, and EL0 accesses to Performance Monitors registers' in the ARMv8 ARM, section D1.

## IMPLEMENTATION DEFINED exception to EL3

This is the layout of the ISS field for exceptions with the following EC values:

- 0b011111, IMPLEMENTATION DEFINED exception to EL3.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMPLEMENTATION DEFINED																								

### IMPLEMENTATION DEFINED, bits [24:0]

IMPLEMENTATION DEFINED.

## Exception from an Instruction Abort

This is the layout of the ISS field for exceptions with the following EC values:

- 0b100000, Instruction Abort from a lower Exception level.
- 0b100001, Instruction Abort taken without a change in Exception level.

24	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	SET	FnV	EA	0	S1PTW	0	IFSC						

### Bits [24:13]

Reserved, RES0.

### SET, bits [12:11]

Synchronous Error Type. When the RAS Extension is implemented and IFSC is 010000, describes the state of the PE after taking the Instruction Abort exception. The possible values of this field are:

SET	Meaning
00	Recoverable error (UER).
01	Restartable error (UEO).
10	Uncontainable error (UC).
11	Corrected error (CE).

### Note

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in an unrecoverable PE state.

This field is RES0 if either:

- The RAS Extension is not implemented.
- The value returned in the IFSC field is not 010000.

### FnV, bit [10]

FAR not Valid, for a Synchronous external abort other than a Synchronous external abort on a translation table walk.

FnV	Meaning
0	FAR is valid.
1	FAR is not valid, and holds an UNKNOWN value.

This field is only valid if the IFSC code is 010000. It is RES0 for all other aborts.

### EA, bit [9]

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

**Bit [8]**

Reserved, RES0.

**S1PTW, bit [7]**

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
0	Fault not on a stage 2 translation for a stage 1 translation table walk.
1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

**Bit [6]**

Reserved, RES0.

**IFSC, bits [5:0]**

Instruction Fault Status Code. Possible values of this field are:

IFSC	Meaning
000000	Address size fault, level 0 of translation or translation table base register
000001	Address size fault, level 1
000010	Address size fault, level 2
000011	Address size fault, level 3
000100	Translation fault, level 0
000101	Translation fault, level 1
000110	Translation fault, level 2
000111	Translation fault, level 3
001001	Access flag fault, level 1
001010	Access flag fault, level 2
001011	Access flag fault, level 3
001101	Permission fault, level 1
001110	Permission fault, level 2
001111	Permission fault, level 3
010000	Synchronous external abort, not on translation table walk
011000	Synchronous parity or ECC error on memory access, not on translation table walk
010100	Synchronous external abort, on translation table walk, level 0
010101	Synchronous external abort, on translation table walk, level 1
010110	Synchronous external abort, on translation table walk, level 2
010111	Synchronous external abort, on translation table walk, level 3
011100	Synchronous parity or ECC error on memory access on translation table walk, level 0
011101	Synchronous parity or ECC error on memory access on translation table walk, level 1
011110	Synchronous parity or ECC error on memory access on translation table walk, level 2
011111	Synchronous parity or ECC error on memory access on translation table walk, level 3
110000	TLB conflict abort

All other values are reserved.

When the RAS Extension is implemented, 011000, 011100, 011101, 011110, and 011111, are reserved.

**Note**

ARMv8.2 requires the implementation of the RAS Extension.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults' in the ARMv8 ARM.

**Note**

Because Access flag faults and Permission faults can only result from a Block or Page translation table descriptor, they cannot occur at level 0.

**Exception from a Data Abort**

This is the layout of the ISS field for exceptions with the following EC values:

- 0b100100, Data Abort from a lower Exception level.
- 0b100101, Data Abort taken without a change in Exception level.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISV	SAS	SSE	SRT				SF	AR	0	SET	FnV	EA	CM	S1PTW	WnR	DFSC								

**ISV, bit [24]**

Instruction syndrome valid. Indicates whether the syndrome information in ISS[23:14] is valid.

ISV	Meaning
0	No valid instruction syndrome. ISS[23:14] are RES0.
1	ISS[23:14] hold a valid instruction syndrome.

This bit is 0 for all faults reported in ESR\_EL2 except the following stage 2 aborts:

- AArch64 loads and stores of a single general-purpose register (including the register specified with 0b11111), including those with Acquire/Release semantics, but excluding Load Exclusive or Store Exclusive and excluding those with writeback.
- AArch32 instructions where the instruction:
  - Is an LDR, LDA, LDRT, LDRSH, LDRSHT, LDRH, LDAH, LDRHT, LDRSB, LDRSBT, LDRB, LDAB, LDRBT, STR, STL, STRT, STRH, STLH, STRHT, STRB, STLB, or STRBT instruction.
  - Is not performing register writeback.
  - Is not using R15 as a source or destination register.

For these cases, ISV is UNKNOWN if the exception was generated in Debug state in memory access mode, and otherwise indicates whether ISS[23:14] hold a valid syndrome.

ISV is 0 for all faults reported in ESR\_EL1 or ESR\_EL3.

When the RAS Extension is implemented, ISV is 0 for any Synchronous external abort.

For ISS reporting, a stage 2 abort on a stage 1 translation table walk does not return a valid instruction syndrome, and therefore ISV is 0 for these aborts.

When the RAS Extension is not implemented, the value of ISV on a Synchronous external abort on a stage 2 translation table walk is IMPLEMENTATION DEFINED.

**SAS, bits [23:22]**

Syndrome Access Size. When ISV is 1, indicates the size of the access attempted by the faulting operation.

SAS	Meaning
00	Byte
01	Halfword
10	Word
11	Doubleword

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

**SSE, bit [21]**

Syndrome Sign Extend. When ISV is 1, for a byte, halfword, or word load operation, indicates whether the data item must be sign extended. For these cases, the possible values of this bit are:

SSE	Meaning
0	Sign-extension not required.
1	Data item must be sign-extended.

For all other operations this bit is 0.

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

### SRT, bits [20:16]

Syndrome Register transfer. When ISV is 1, the register number of the Rt operand of the faulting instruction. If the exception was taken from an Exception level that is using AArch32 then this is the AArch64 view of the register. See 'Mapping of the general-purpose registers between the Execution states' in the ARMv8 ARM, section D1.20.1.

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

### SF, bit [15]

Width of the register accessed by the instruction is Sixty-Four. When ISV is 1, the possible values of this bit are:

SF	Meaning
0	Instruction loads/stores a 32-bit wide register.
1	Instruction loads/stores a 64-bit wide register.

#### Note

This field specifies the register width identified by the instruction, not the Execution state.

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

### AR, bit [14]

Acquire/Release. When ISV is 1, the possible values of this bit are:

AR	Meaning
0	Instruction did not have acquire/release semantics.
1	Instruction did have acquire/release semantics.

This field is UNKNOWN when the value of ISV is UNKNOWN.

This field is RES0 when the value of ISV is 0.

### Bit [13]

Reserved, RES0.

### SET, bits [12:11]

Synchronous Error Type. When the RAS Extension is implemented and DFSC is 010000, describes the state of the PE after taking the Data Abort exception. The possible values of this field are:

SET	Meaning
00	Recoverable error (UER).
01	Restartable error (UEO).
10	Uncontainable error (UC).
11	Corrected error (CE).

**Note**

Software can use this information to determine what recovery might be possible. Taking a synchronous External Abort exception might result in an unrecoverable PE state.

This field is RES0 if either:

- The RAS Extension is not implemented.
- The value returned in the DFSC field is not 010000.

**FnV, bit [10]**

FAR not Valid, for a Synchronous external abort other than a Synchronous external abort on a translation table walk.

FnV	Meaning
0	FAR is valid.
1	FAR is not valid, and holds an UNKNOWN value.

This field is valid only if the DFSC code is 010000. It is RES0 for all other aborts.

**EA, bit [9]**

External abort type. This bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

**CM, bit [8]**

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

CM	Meaning
0	The Data Abort was not generated by the execution of one of the system instructions identified in the description of value 1.
1	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The <a href="#">DC ZVA</a> instruction is not classified as a cache maintenance instruction, and therefore its execution cannot cause this field to be set to 1.

**S1PTW, bit [7]**

For a stage 2 fault, indicates whether the fault was a stage 2 fault on an access made for a stage 1 translation table walk:

S1PTW	Meaning
0	Fault not on a stage 2 translation for a stage 1 translation table walk.
1	Fault on the stage 2 translation of an access for a stage 1 translation table walk.

For any abort other than a stage 2 fault this bit is RES0.

**WnR, bit [6]**

Write not Read. Indicates whether a synchronous abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location. The possible values of this bit are:

WnR	Meaning
0	Abort caused by an instruction reading from a memory location.
1	Abort caused by an instruction writing to a memory location.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction that both reads and writes from a memory location, this bit is set to 0 if a read of the address specified by the instruction would have generated the fault is being reported, otherwise it is set to 1. The architecture permits, but does

not require, a relaxation of this requirement such that for all stage 2 aborts on stage 1 translation table walks for atomic instructions, the WnR bit is always 0.

This field is UNKNOWN for:

- An external abort.
- A fault reported using the DFSC value of 0b110101 that indicates an atomic instruction that is unsupported by the memory type.

## DFSC, bits [5:0]

Data Fault Status Code. Possible values of this field are:

DFSC	Meaning
000000	Address size fault, level 0 of translation or translation table base register
000001	Address size fault, level 1
000010	Address size fault, level 2
000011	Address size fault, level 3
000100	Translation fault, level 0
000101	Translation fault, level 1
000110	Translation fault, level 2
000111	Translation fault, level 3
001001	Access flag fault, level 1
001010	Access flag fault, level 2
001011	Access flag fault, level 3
001101	Permission fault, level 1
001110	Permission fault, level 2
001111	Permission fault, level 3
010000	Synchronous external abort, not on translation table walk
011000	Synchronous parity or ECC error on memory access, not on translation table walk
010100	Synchronous external abort, on translation table walk, level 0
010101	Synchronous external abort, on translation table walk, level 1
010110	Synchronous external abort, on translation table walk, level 2
010111	Synchronous external abort, on translation table walk, level 3
011100	Synchronous parity or ECC error on memory access on translation table walk, level 0
011101	Synchronous parity or ECC error on memory access on translation table walk, level 1
011110	Synchronous parity or ECC error on memory access on translation table walk, level 2
011111	Synchronous parity or ECC error on memory access on translation table walk, level 3
100001	Alignment fault
110000	TLB conflict abort
110001	Unsupported atomic hardware update fault, if the implementation includes ARMv8.1-TTHM. Otherwise reserved.
110100	IMPLEMENTATION DEFINED fault (Lockdown)
110101	IMPLEMENTATION DEFINED fault (Unsupported Exclusive or Atomic access)
111101	Section Domain Fault, used only for faults reported in the <a href="#">PAR_EL1</a>
111110	Page Domain Fault, used only for faults reported in the <a href="#">PAR_EL1</a>

All other values are reserved.

When the RAS Extension is implemented, 011000, 011100, 011101, 011110, and 011111, are reserved.

For more information about the lookup level associated with a fault, see 'The level associated with MMU faults' in the ARMv8 ARM.

### Note

Because Access flag faults and Permission faults can only result from a Block or Page translation table descriptor, they cannot occur at level 0.



## Exception from a trapped floating-point exception

This is the layout of the ISS field for exceptions with the following EC values:

- 0b101000, Trapped floating-point exception taken from AArch32 state.
- 0b101100, Trapped floating-point exception taken from AArch64 state.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	TFV	0	0	0	0	0	0	0	0	0	0	0	0	VECITR	IDF	0	0	IXF	UFF	OFF	DZF	IOF		

### Bit [24]

Reserved, RES0.

### TFV, bit [23]

Trapped Fault Valid bit. Indicates whether the IDF, IXF, UFF, OFF, DZF, and IOF bits hold valid information about trapped floating-point exceptions. The possible values of this bit are:

TFV	Meaning
0	The IDF, IXF, UFF, OFF, DZF, and IOF bits do not hold valid information about trapped floating-point exceptions and are UNKNOWN.
1	One or more floating-point exceptions occurred during an operation performed while executing the reported instruction. The IDF, IXF, UFF, OFF, DZF, and IOF bits indicate trapped floating-point exceptions that occurred. For more information see 'Floating-point exception traps' in the ARMv8 ARM, section D1.13.4.

It is IMPLEMENTATION DEFINED whether this field is set to 0 on an exception generated by a trapped floating point exception from a vector instruction.

### Note

This is not a requirement. Implementations can set this field to 1 on a trapped floating-point exception from a vector instruction and return valid information in the {IDF, IXF, UFF, OFF, DZF, IOF} fields.

### Bits [22:11]

Reserved, RES0.

### VECITR, bits [10:8]

For a trapped floating-point exception from an instruction executed in AArch32 state this field is RES1.

For a trapped floating-point exception from an instruction executed in AArch64 state this field is UNKNOWN.

### IDF, bit [7]

Input Denormal floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

IDF	Meaning
0	Input denormal floating-point exception has not occurred.
1	Input denormal floating-point exception occurred during execution of the reported instruction.

### Bits [6:5]

Reserved, RES0.

**IXF, bit [4]**

Inexact floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

IXF	Meaning
0	Inexact floating-point exception has not occurred.
1	Inexact floating-point exception occurred during execution of the reported instruction.

**UFF, bit [3]**

Underflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

UFF	Meaning
0	Underflow floating-point exception has not occurred.
1	Underflow floating-point exception occurred during execution of the reported instruction.

**OFF, bit [2]**

Overflow floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

OFF	Meaning
0	Overflow floating-point exception has not occurred.
1	Overflow floating-point exception occurred during execution of the reported instruction.

**DZF, bit [1]**

Divide by Zero floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

DZF	Meaning
0	Divide by Zero floating-point exception has not occurred.
1	Divide by Zero floating-point exception occurred during execution of the reported instruction.

**IOF, bit [0]**

Invalid Operation floating-point exception trapped bit. If the TFV field is 0, this bit is UNKNOWN. Otherwise, the possible values of this bit are:

IOF	Meaning
0	Invalid Operation floating-point exception has not occurred.
1	Invalid Operation floating-point exception occurred during execution of the reported instruction.

In an implementation that supports the trapping of floating-point exceptions:

- From an Exception level using AArch64, the [FPCR](#).{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.
- From an Exception level using AArch32, the [FPSCR](#).{IDE, IXE, UFE, OFE, DZE, IOE} bits enable each of the floating-point exception traps.

**SError interrupt**

This is the layout of the ISS field for exceptions with the following EC values:

- 0b101111, SError interrupt.

	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	<b>IDS</b>	0	0	0	0	0	0	0	0	0	0	<b>IESB</b>		<b>AET</b>		<b>EA</b>	0	0	0			<b>DFSC</b>			

**IDS, bit [24]**

IMPLEMENTATION DEFINED syndrome. Possible values of this bit are:

IDS	Meaning
0	Bits[23:0] of the ISS field are defined in this description.
<b>Note</b> If the RAS Extension is not implemented, this means that bits[23:0] of the ISS field are RES0.	
1	Bits[23:0] of the ISS field holds IMPLEMENTATION DEFINED syndrome information that can be used to provide additional information about the SError interrupt.

**Note**

This field was previously called ISV.

**Bits [23:14]**

Reserved, RES0.

**IESB, bit [13]**

Implicit Error Synchronization Barrier.

This field is part of the RAS Extension in an ARMv8.2 implementation.

The possible values of this field are:

IESB	Meaning
0	The SError interrupt was either not synchronized by the implicit ErrorSynchronizationBarrier() or not taken immediately.
1	The SError interrupt was synchronized by the implicit ErrorSynchronizationBarrier() and taken immediately.

This field is RES0 if either:

- The implementation is an ARMv8.0 or ARMv8.1 implementation.
- The value returned in the DFSC field is not 010001.

**Note**

ARMv8.2 requires the implementation of the RAS Extension.

**AET, bits [12:10]**

Asynchronous Error Type.

When the RAS Extension is implemented and DFSC is 010001, describes the state of the PE after taking the SError interrupt exception. The possible values of this field are:

AET	Meaning
000	Uncontainable error (UC).
001	Unrecoverable error (UEU).
010	Restartable error (UEO).
011	Recoverable error (UER).
110	Corrected error (CE).

All other values are reserved.

If multiple errors are taken as a single SError interrupt exception, the overall state of the PE is reported. For example, if both a Recoverable and Unrecoverable error occurred, the state is Unrecoverable.

**Note**

Software can use this information to determine what recovery might be possible. The recovery software must also examine any implemented fault records to determine the location and extent of the error.

This field is RES0 if either:

- The RAS Extension is not implemented.
- The value returned in the DFSC field is not 010001.

**Note**

ARMv8.2 requires the implementation of the RAS Extension.

**EA, bit [9]**

External abort type. When the RAS Extension is implemented, this bit can provide an IMPLEMENTATION DEFINED classification of External aborts.

For any abort other than an External abort this bit returns a value of 0.

This field is RES0 if either:

- The RAS Extension is not implemented.
- The value returned in the DFSC field is not 010001.

**Note**

ARMv8.2 requires the implementation of the RAS Extension.

**Bits [8:6]**

Reserved, RES0.

**DFSC, bits [5:0]**

Data Fault Status Code. When the RAS Extension is implemented, possible values of this field are:

DFSC	Meaning
000000	Uncategorized.
010001	Asynchronous SError interrupt.

All other values are reserved.

If the RAS Extension is not implemented, this field is RES0.

**Note**

ARMv8.2 requires the implementation of the RAS Extension.

**Exception from a Breakpoint or Vector Catch debug exception**

This is the layout of the ISS field for exceptions with the following EC values:

- 0b110000, Breakpoint exception from a lower Exception level.
- 0b110001, Breakpoint exception taken without a change in Exception level.
- 0b111010, Vector Catch exception from AArch32 state.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IFSC					

**Bits [24:6]**

Reserved, RES0.

**IFSC, bits [5:0]**

Instruction Fault Status Code. This field is set to 0b100010, to indicate a Debug exception.

For more information about generating these exceptions:

- For exceptions from AArch64, see 'Breakpoint exceptions' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).
- For exceptions from AArch32, see 'Breakpoint exceptions' in the ARMv8 ARM, section G2 (AArch32 Self-hosted Debug) and 'Vector Catch exceptions' in the ARMv8 ARM, section G2.

**Exception from a Software Step exception**

This is the layout of the ISS field for exceptions with the following EC values:

- 0b110010, Software Step exception from a lower Exception level.
- 0b110011, Software Step exception taken without a change in Exception level.

	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISV	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EX						
																							IFSC		

**ISV, bit [24]**

Instruction syndrome valid. Indicates whether the EX bit, ISS[6], is valid, as follows:

ISV	Meaning
0	EX bit is RES0.
1	EX bit is valid.

See the EX bit description for more information.

**Bits [23:7]**

Reserved, RES0.

**EX, bit [6]**

Exclusive operation. If the ISV bit is set to 1, this bit indicates whether a Load-Exclusive instruction was stepped.

EX	Meaning
0	An instruction other than a Load-Exclusive instruction was stepped.
1	A Load-Exclusive instruction was stepped.

If the ISV bit is set to 0, this bit is RES0, indicating no syndrome data is available.

**IFSC, bits [5:0]**

Instruction Fault Status Code. This field is set to 0b100010, to indicate a Debug exception.

For more information about generating these exceptions, see 'Software Step exceptions' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).

**Exception from a Watchpoint exception**

This is the layout of the ISS field for exceptions with the following EC values:

- 0b110100, Watchpoint exception from a lower Exception level.
- 0b110101, Watchpoint exception taken without a change in Exception level.

	24	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	CM	0	WnR						
																							DFSC		

**Bits [24:9]**

Reserved, RES0.

**CM, bit [8]**

Cache maintenance. Indicates whether the Data Abort came from a cache maintenance or address translation instruction:

CM	Meaning
0	The Data Abort was not generated by the execution of one of the system instructions identified in the description of value 1.
1	The Data Abort was generated by either the execution of a cache maintenance instruction or by a synchronous fault on the execution of an address translation instruction. The <a href="#">DC ZVA</a> instruction is not classified as a cache maintenance instruction, and therefore its execution cannot cause this field to be set to 1.

**Bit [7]**

Reserved, RES0.

**WnR, bit [6]**

Write not Read. Indicates whether the abort was caused by an instruction writing to a memory location, or by an instruction reading from a memory location. The possible values of this bit are:

WnR	Meaning
0	Abort caused by an instruction reading from a memory location.
1	Abort caused by an instruction writing to a memory location.

For faults on cache maintenance and address translation instructions, this bit always returns a value of 1.

For faults from an atomic instruction, this field is set to 0 if a read of the location would have generated a fault, otherwise it is set to 1.

**DFSC, bits [5:0]**

Data Fault Status Code. This field is set to 0b100010, to indicate a Debug exception.

For more information about generating these exceptions, see 'Watchpoint exceptions' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).

**Exception from execution of a Breakpoint instruction**

This is the layout of the ISS field for exceptions with the following EC values:

- 0b111000, BKPT instruction execution in AArch32 state.
- 0b111100, BRK instruction execution in AArch64 state.

24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0																

[Comment](#)

**Bits [24:16]**

Reserved, RES0.

**Comment, bits [15:0]**

Set to the instruction comment field value, zero extended as necessary. For the AArch32 BKPT instructions, the comment field is described as the immediate field.

For more information about generating these exceptions, see 'Breakpoint instruction exceptions' in the ARMv8 ARM, section D2 (AArch64 Self-hosted Debug).



# FAR\_EL1, Fault Address Register (EL1)

The FAR\_EL1 characteristics are:

## Purpose

Holds the faulting Virtual Address for all synchronous Instruction or Data Abort, PC alignment fault and Watchpoint exceptions that are taken to EL1.

This register is part of the Exception and fault handling registers functional group.

## Configuration

AArch64 System register FAR\_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DFAR \(NS\)](#).

AArch64 System register FAR\_EL1 bits [63:32] are architecturally mapped to AArch32 System register [IFAR \(NS\)](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

FAR\_EL1 is a 64-bit register.

## Field descriptions

The FAR\_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
Faulting Virtual Address for synchronous exceptions taken to EL1																															
Faulting Virtual Address for synchronous exceptions taken to EL1																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Faulting Virtual Address for synchronous exceptions taken to EL1. Exceptions that set the FAR\_EL1 are Instruction Aborts (EC 0x20 or 0x21), Data Aborts (EC 0x24 or 0x25), PC alignment faults (EC 0x22), and Watchpoints (EC 0x34 or 0x35). [ESR\\_EL1](#).EC holds the EC value for the exception.

For a synchronous external abort, if the VA that generated the abort was from an address range for which TCR\_ELx.TBI{<0|1>} == 1 for the translation regime in use when the abort was generated, then the top eight bits of FAR\_EL1 are UNKNOWN.

For a synchronous external abort other than a synchronous external abort on a translation table walk, this field is valid only if [ESR\\_EL1](#).FnV is 0, and the FAR\_EL1 is UNKNOWN if [ESR\\_EL1](#).FnV is 1.

For all other exceptions taken to EL1, the FAR\_EL1 is UNKNOWN.

If a memory fault that sets FAR\_EL1 is generated from a data cache maintenance or DC ZVA instruction, this field holds the address specified in the register argument of the instruction.

If the exception that updates FAR\_EL1 is taken from an Exception level that is using AArch32, the top 32 bits are all zero, unless the faulting address is generated by a load or store instruction that sequentially increments from address 0xFFFFFFFF. This is an UNPREDICTABLE condition, and in this case the upper 32-bits are set to 0x00000001.

For a Data Abort or Watchpoint exception, if address tagging is enabled for the address accessed by the data access that caused the exception, then this field includes the tag. For more information about address tagging, see 'Address tagging in AArch64 state' in the ARMv8 ARM.

---

### Note

---



The address held in this field is an address accessed by the instruction fetch or data access that caused the exception that gave rise to the instruction or data abort. It is the lower address that gave rise to the fault. Where different faults from different addresses arise from the same instruction, such as for an instruction that loads or stores a mis-aligned address that crosses a page boundary, the architecture does not prioritize between those different faults.

FAR\_EL1 is made UNKNOWN on an exception return from EL1.

## Accessing the FAR\_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
FAR_EL1	11	000	0110	0000	000
FAR_EL12	11	101	0110	0000	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
FAR_EL1	x	x	0	-	RW	n/a	RW
FAR_EL1	0	0	1	-	RW	RW	RW
FAR_EL1	0	1	1	-	n/a	RW	RW
FAR_EL1	1	0	1	-	RW	<a href="#">FAR_EL2</a>	RW
FAR_EL1	1	1	1	-	n/a	<a href="#">FAR_EL2</a>	RW
FAR_EL12	x	x	0	-	-	n/a	-
FAR_EL12	0	0	1	-	-	-	-
FAR_EL12	0	1	1	-	n/a	-	-
FAR_EL12	1	0	1	-	-	RW	RW
FAR_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic FAR\_EL1 or FAR\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

## 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.

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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# FAR\_EL2, Fault Address Register (EL2)

The FAR\_EL2 characteristics are:

## Purpose

Holds the faulting Virtual Address for all synchronous Instruction or Data Abort, PC alignment fault and Watchpoint exceptions that are taken to EL2.

This register is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.

## Configuration

AArch64 System register FAR\_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HDFAR](#).

AArch64 System register FAR\_EL2 bits [63:32] are architecturally mapped to AArch32 System register [HIFAR](#).

AArch64 System register FAR\_EL2 bits [31:0] are architecturally mapped to AArch32 System register [DFAR \(S\)](#) when EL2 is implemented.

AArch64 System register FAR\_EL2 bits [63:32] are architecturally mapped to AArch32 System register [IFAR \(S\)](#) when EL2 is implemented.

If EL2 is not implemented, this register is RES0 from EL3.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

FAR\_EL2 is a 64-bit register.

## Field descriptions

The FAR\_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
Faulting Virtual Address for synchronous exceptions taken to EL2																															
Faulting Virtual Address for synchronous exceptions taken to EL2																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Faulting Virtual Address for synchronous exceptions taken to EL2. Exceptions that set the FAR\_EL2 are Instruction Aborts (EC 0x20 or 0x21), Data Aborts (EC 0x24 or 0x25), PC alignment faults (EC 0x22), and Watchpoints (EC 0x34 or 0x35). [ESR\\_EL2](#).EC holds the EC value for the exception.

For a synchronous external abort, if the VA that generated the abort was from an address range for which TCR\_ELx.TBI{<0|1>} = 1 for the translation regime in use when the abort was generated, then the top eight bits of FAR\_EL2 are UNKNOWN.

For a synchronous external abort other than a synchronous external abort on a translation table walk, this field is valid only if [ESR\\_EL2](#).FnV is 0, and the FAR\_EL2 is UNKNOWN if [ESR\\_EL2](#).FnV is 1.

For all other exceptions taken to EL2, the FAR\_EL2 is UNKNOWN.

If a memory fault that sets FAR\_EL2 is generated from a data cache maintenance or DC ZVA instruction, this field holds the address specified in the register argument of the instruction.

If the exception that updates FAR\_EL2 is taken from an Exception level that is using AArch32, the top 32-bits are all zero, unless the faulting address is generated by a load or store instruction that sequentially increments from address 0xFFFFFFFF. This is an UNPREDICTABLE condition, and in this case the upper 32-bits are set to 0x00000001.

For a Data Abort or Watchpoint exception, if address tagging is enabled for the address accessed by the data access that caused the exception, then this field includes the tag. For more information about address tagging, see 'Address tagging in AArch64 state' in the ARMv8 ARM.

## Note

The address held in this field is an address accessed by the instruction fetch or data access that caused the exception that gave rise to the instruction or data abort. It is the lower address that gave rise to the fault. Where different faults from different addresses arise from the same instruction, such as for an instruction that loads or stores a mis-aligned address that crosses a page boundary, the architecture does not prioritize between those different faults.

FAR\_EL2 is made UNKNOWN on an exception return from EL2.

## Accessing the FAR\_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
FAR_EL2	11	100	0110	0000	000
FAR_EL1	11	000	0110	0000	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
FAR_EL2	x	x	0	-	-	n/a	RW
FAR_EL2	0	0	1	-	-	RW	RW
FAR_EL2	0	1	1	-	n/a	RW	RW
FAR_EL2	1	0	1	-	-	RW	RW
FAR_EL2	1	1	1	-	n/a	RW	RW
FAR_EL1	x	x	0	-	<a href="#">FAR_EL1</a>	n/a	<a href="#">FAR_EL1</a>
FAR_EL1	0	0	1	-	<a href="#">FAR_EL1</a>	<a href="#">FAR_EL1</a>	<a href="#">FAR_EL1</a>
FAR_EL1	0	1	1	-	n/a	<a href="#">FAR_EL1</a>	<a href="#">FAR_EL1</a>
FAR_EL1	1	0	1	-	<a href="#">FAR_EL1</a>	RW	<a href="#">FAR_EL1</a>
FAR_EL1	1	1	1	-	n/a	RW	<a href="#">FAR_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic FAR\_EL2 or FAR\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

# FAR\_EL3, Fault Address Register (EL3)

The FAR\_EL3 characteristics are:

## Purpose

Holds the faulting Virtual Address for all synchronous Instruction or Data Abort and PC alignment fault exceptions that are taken to EL3.

This register is part of the Exception and fault handling registers functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

FAR\_EL3 is a 64-bit register.

## Field descriptions

The FAR\_EL3 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
Faulting Virtual Address for synchronous exceptions taken to EL3																															
Faulting Virtual Address for synchronous exceptions taken to EL3																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Faulting Virtual Address for synchronous exceptions taken to EL3. Exceptions that set the FAR\_EL3 are Instruction Aborts (EC 0x20 or 0x21), Data Aborts (EC 0x24 or 0x25), and PC alignment faults (EC 0x22). [ESR\\_EL3](#).EC holds the EC value for the exception.

For a synchronous external abort, if the VA that generated the abort was from an address range for which TCR\_ELx.TBI{<0|1>} == 1 for the translation regime in use when the abort was generated, then the top eight bits of FAR\_EL3 are UNKNOWN.

For a synchronous external abort other than a synchronous external abort on a translation table walk, this field is valid only if [ESR\\_EL3](#).FnV is 0, and the FAR\_EL3 is UNKNOWN if [ESR\\_EL3](#).FnV is 1.

For all other exceptions taken to EL3, the FAR\_EL3 is UNKNOWN.

If a memory fault that sets FAR\_EL3 is generated from a data cache maintenance or DC ZVA instruction, this field holds the address specified in the register argument of the instruction.

If the exception that updates FAR\_EL3 is taken from an EL using AArch32, the top 32-bits are all zero, unless the faulting address is generated by a load or store instruction that sequentially increments from address 0xffffffff. This is an UNPREDICTABLE condition, and in this case the upper 32-bits are set to 0x00000001.

For a Data Abort or Watchpoint exception, if address tagging is enabled for the address accessed by the data access that caused the exception, then this field includes the tag. For more information about address tagging, see 'Address tagging in AArch64 state' in the ARMv8 ARM.

---

### Note

The address held in this register is an address accessed by the instruction fetch or data access that caused the exception that actually gave rise to the instruction or data abort. It is the lowest address that gave rise to the fault. Where different faults from different addresses arise from the same instruction, such as for an instruction that loads or stores a mis-aligned address that crosses a page boundary, the architecture does not prioritize between those different faults.

---

FAR\_EL3 is made UNKNOWN on an exception return from EL3.

## Accessing the FAR\_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
FAR_EL3	11	110	0110	0000	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
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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# FPCR, Floating-point Control Register

The FPCR characteristics are:

## Purpose

Controls floating-point behavior.

This register is part of:

- The Special-purpose registers functional group.
- The Floating-point registers functional group.

## Configuration

The named fields in this register map to the equivalent fields in the AArch32 [FPSCR](#).

It is IMPLEMENTATION DEFINED whether the Len and Stride fields can be programmed to non-zero values, which will cause some AArch32 floating-point instruction encodings to be UNDEFINED, or whether these fields are RAZ.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

FPCR is a 32-bit register.

## Field descriptions

The FPCR 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	AHP	DN	FZ	RMode	Stride	FZ16	Len	IDE	0	0	IXE	UFE	OF	EDZE	IOE	0	0	0	0	0	0	0	0	0	0	0	0

### Bits [31:27]

Reserved, RES0.

### AHP, bit [26]

Alternative half-precision control bit:

AHP	Meaning
0	IEEE half-precision format selected.
1	Alternative half-precision format selected.

This bit is only used for conversions between half-precision floating-point and other floating-point formats.

The data-processing instructions added as part of the ARMv8.2-FP16 extension always use the IEEE half-precision format, and ignore the value of this bit.

### DN, bit [25]

Default NaN mode control bit:

DN	Meaning
0	NaN operands propagate through to the output of a floating-point operation.
1	Any operation involving one or more NaNs returns the Default NaN.

The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.

### FZ, bit [24]

Flush-to-zero mode control bit:

FZ	Meaning
0	Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.
1	Flush-to-zero mode enabled.

The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.

This bit has no effect on half-precision calculations.

### RMode, bits [23:22]

Rounding Mode control field. The encoding of this field is:

RMode	Meaning
00	Round to Nearest (RN) mode
01	Round towards Plus Infinity (RP) mode
10	Round towards Minus Infinity (RM) mode
11	Round towards Zero (RZ) mode.

The specified rounding mode is used by both scalar and Advanced SIMD floating-point instructions.

### Stride, bits [21:20]

This field has no function in AArch64 state, and non-zero values are ignored during execution in AArch64 state. It is included only for context saving and restoration of the AArch32 [FPSCR](#).Stride field.

### FZ16, bit [19]

In ARMv8.2:

When ARMv8.2-FP16 is implemented, flush-to-zero mode control bit on half-precision data-processing instructions:

FZ16	Meaning
0	Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.
1	Flush-to-zero mode enabled.

The value of this bit applies to both scalar and Advanced SIMD floating-point half-precision calculations. A half-precision floating-point number that is flushed to zero as a result of the value of the FZ16 bit does not generate an Input Denormal exception.

When ARMv8.2-FP16 is not implemented, this bit is RES0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

### Len, bits [18:16]

This field has no function in AArch64 state, and non-zero values are ignored during execution in AArch64 state. It is included only for context saving and restoration of the AArch32 [FPSCR](#).Len field.

### IDE, bit [15]

Input Denormal floating-point exception trap enable. Possible values are:



IDE	Meaning
0	Untrapped exception handling selected. If the floating-point exception occurs then the <a href="#">FPSR.IDC</a> bit is set to 1.
1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the <a href="#">FPSR.IDC</a> bit. The trap handling software can decide whether to set the <a href="#">FPSR.IDC</a> bit to 1.

The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.

If the implementation does not support this exception, this bit is RES0.

#### Bits [14:13]

Reserved, RES0.

#### IXE, bit [12]

Inexact floating-point exception trap enable. Possible values are:

IXE	Meaning
0	Untrapped exception handling selected. If the floating-point exception occurs then the <a href="#">FPSR.IXC</a> bit is set to 1.
1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the <a href="#">FPSR.IXC</a> bit. The trap handling software can decide whether to set the <a href="#">FPSR.IXC</a> bit to 1.

The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.

If the implementation does not support this exception, this bit is RES0.

#### UFE, bit [11]

Underflow floating-point exception trap enable. Possible values are:

UFE	Meaning
0	Untrapped exception handling selected. If the floating-point exception occurs then the <a href="#">FPSR.UFC</a> bit is set to 1.
1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the <a href="#">FPSR.UFC</a> bit. The trap handling software can decide whether to set the <a href="#">FPSR.UFC</a> bit to 1.

The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.

If the implementation does not support this exception, this bit is RES0.

#### OFE, bit [10]

Overflow floating-point exception trap enable. Possible values are:

OFE	Meaning
0	Untrapped exception handling selected. If the floating-point exception occurs then the <a href="#">FPSR.OFC</a> bit is set to 1.
1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the <a href="#">FPSR.OFC</a> bit. The trap handling software can decide whether to set the <a href="#">FPSR.OFC</a> bit to 1.

The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.

If the implementation does not support this exception, this bit is RES0.

#### DZE, bit [9]

Divide by Zero floating-point exception trap enable. Possible values are:

DZE	Meaning
0	Untrapped exception handling selected. If the floating-point exception occurs then the <a href="#">FPSR.DZC</a> bit is set to 1.
1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the <a href="#">FPSR.DZC</a> bit. The trap handling software can decide whether to set the <a href="#">FPSR.DZC</a> bit to 1.

The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.

If the implementation does not support this exception, this bit is RES0.

## IOE, bit [8]

Invalid Operation floating-point exception trap enable. Possible values are:

IOE	Meaning
0	Untrapped exception handling selected. If the floating-point exception occurs then the <a href="#">FPSR.IOE</a> bit is set to 1.
1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the <a href="#">FPSR.IOE</a> bit. The trap handling software can decide whether to set the <a href="#">FPSR.IOE</a> bit to 1.

The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.

If the implementation does not support this exception, this bit is RES0.

## Bits [7:0]

Reserved, RES0.

# Accessing the FPCR

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
FPCR	11	011	0100	0100	000

## 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.

When HCR\_EL2.E2H==0 :

- If [CPACR\\_EL1](#).FPEN==00, accesses to this register from EL0 and EL1 are trapped to EL1.
- If [CPACR\\_EL1](#).FPEN==01, accesses to this register from EL0 are trapped to EL1.
- If [CPACR\\_EL1](#).FPEN==10, accesses to this register from EL0 and EL1 are trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [CPTR\\_EL2](#).TFP==1, Non-secure accesses to this register from EL0, EL1, and EL2 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 [CPACR\\_EL1](#).FPEN==00, Non-secure accesses to this register from EL0 and EL1 are trapped to EL1.
- If [CPACR\\_EL1](#).FPEN==01, Non-secure accesses to this register from EL0 are trapped to EL1.
- If [CPACR\\_EL1](#).FPEN==10, Non-secure accesses to this register from EL0 and EL1 are trapped to EL1.
- If [CPTR\\_EL2](#).FPEN==00, Non-secure accesses to this register from EL0, EL1, and EL2 are trapped to EL2.
- If [CPTR\\_EL2](#).FPEN==10, Non-secure accesses to this register from EL0, EL1, and EL2 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 [CPTR\\_EL2](#).FPEN==00, Non-secure accesses to this register from EL0 and EL2 are trapped to EL2.
- If [CPTR\\_EL2](#).FPEN==01, Non-secure accesses to this register from EL0 are trapped to EL2.
- If [CPTR\\_EL2](#).FPEN==10, Non-secure accesses to this register from EL0 and EL2 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [CPTR\\_EL3](#).TFP==1, accesses to this register from EL0, EL1, EL2, and EL3 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# FPEXC32\_EL2, Floating-Point Exception Control register

The FPEXC32\_EL2 characteristics are:

## Purpose

Allows access to the AArch32 register [FPEXC](#) from AArch64 state only. Its value has no effect on execution in AArch64 state.

This register is part of the Floating-point registers functional group.

## Configuration

AArch64 System register FPEXC32\_EL2 is architecturally mapped to AArch32 System register [FPEXC](#).

If EL1 cannot use AArch32, this register is UNDEFINED.

If EL2 is not implemented but EL3 is implemented, and EL1 is capable of using AArch32, then this register is not RES0.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

FPEXC32\_EL2 is a 32-bit register.

## Field descriptions

The FPEXC32\_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
EX	EN	DEX	FP2V	VV	TFV	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VEC	ITR	IDF	0	0	IXF	UFF	OFF	DZF	IOF

### EX, bit [31]

Exception bit. In ARMv8, this bit is RAZ/WI.

### EN, bit [30]

Enables access to the Advanced SIMD and floating-point functionality from all Exception levels, except that setting this field to 0 does not disable the following:

- VMSR accesses to the [FPEXC](#) or [FPSID](#).
- VMRS accesses from the [FPEXC](#), [FPSID](#), [MVFR0](#), [MVFR1](#), or [MVFR2](#).

EN	Meaning
0	Accesses to the <a href="#">FPSCR</a> , and any of the SIMD and floating-point registers Q0-Q15, including their views as D0-D31 registers or S0-S31 registers, are UNDEFINED at all Exception levels.
1	This control permits access to the Advanced SIMD and floating-point functionality at all Exception levels.

Execution of floating-point and Advanced SIMD instructions in AArch32 state can be disabled or trapped by the following controls:

- [CPACR](#).cp10, or, if executing at EL0, [CPACR\\_EL1](#).FPEN.
- FPEXC.EN.
- If executing in Non-secure state:
  - [HCPTR](#).TCP10, or if EL2 is using AArch64, [CPTR\\_EL2](#).TFP.
  - [NSACR](#).cp10, or if EL3 is using AArch64, [CPTR\\_EL3](#).TFP.
- For Advanced SIMD instructions only:
  - CPACR.ASEDIS.
  - If executing in Non-secure state, [HCPTR](#).TASE and [NSACR](#).NSTRCDIS.

See the descriptions of the controls for more information.

## Note

When executing at EL0 using AArch32:

- If EL1 is using AArch64 then behavior is as if the value of FPEXC.EN is 1.
- In Non-secure state, if EL2 is using AArch64 and the value of [HCR\\_EL2](#).{RW, TGE} is {1, 1} then behavior is as if the value of FPEXC.EN is 1.
- In Non-secure state, if EL2 is using AArch64 and the value of [HCR\\_EL2](#).{RW, TGE} is {0, 1} then it is IMPLEMENTATION DEFINED whether the behavior is:
  - As if the value of FPEXC.EN is 1.
  - Determined by the value of FPEXC32\_EL2.EN, as described in this field description. However, ARM deprecates using the value of FPEXC32\_EL2.EN to determine behavior.

## DEX, bit [29]

Defined synchronous exception on floating-point execution.

This field identifies whether a synchronous exception generated by the attempted execution of an instruction was generated by an unallocated encoding. The instruction must be in the encoding space that is identified by the pseudocode function ExecutingCP10or11Instr() returning TRUE. This field also indicates whether the FPEXC32\_EL2.TFV field is valid.

The meaning of this bit is:

DEX	Meaning
0	The exception was generated by the attempted execution of an unallocated instruction in the encoding space that is identified by the pseudocode function ExecutingCP10or11Instr(). If FPEXC32_EL2.TFV is RW then it is invalid and UNKNOWN. If FPEXC32_EL2.{IDF, IXF, UFF, OFF, DZF, IOF} are RW then they are invalid and UNKNOWN.
1	The exception was generated during the execution of an unallocated encoding. FPEXC32_EL2.TFV is valid and indicates the cause of the exception.

On an exception that sets this bit to 1 the exception-handling routine must clear this bit to 0.

On an implementation that both does not support trapping of floating-point exceptions and implements the AArch32 [FPSCR](#).{Stride, Len} fields as RAZ, this bit is RES0.

## FP2V, bit [28]

FPINST2 instruction valid bit. In ARMv8, this bit is RES0.

## VV, bit [27]

VECITR valid bit. In ARMv8, this bit is RES0.

## TFV, bit [26]

Trapped Fault Valid bit. Valid only when the value of FPEXC.DEX is 1. When valid, it indicates the cause of the exception and therefore whether the FPEXC.{IDF, IXF, UFF, OFF, DZF, IOF} bits are valid.

TFV	Meaning
0	The exception was caused by the execution of a floating-point VABS, VADD, VDIV, VFMA, VFMS, VFNMA, VFNMS, VMLA, VMLS, VMOV, VMUL, VNEG, VNMLA, VNMLS, VNMUL, VSQRT, or VSUB instruction when one or both of <a href="#">FPSCR</a> .{Stride, Len} was non-zero. If the FPEXC.{IDF, IXF, UFF, OFF, DZF, IOF} bits are RW then they are invalid and UNKNOWN.
1	FPEXC.{IDF, IXF, UFF, OFF, DZF, IOF} indicate the presence of trapped floating-point exceptions that had occurred at the time of the exception. Bits are set for all trapped exceptions that had occurred at the time of the exception.

This bit returns a status value and ignores writes.

When the value of FPEXC.DEX is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions this bit is RAZ/WI.

On an implementation that supports the trapping of floating-point exceptions and implements [FPSCR](#).{Stride, Len} as RAZ, this bit is RAO/WI.

#### Bits [25:11]

Reserved, RES0.

#### VECITR, bits [10:8]

Vector iteration count. In ARMv8, this field is RES1.

#### IDF, bit [7]

Input Denormal trapped exception bit. Valid only when the value of FPEXC.TFV is 1. When valid, it indicates whether an Input Denormal exception occurred while [FPSCR](#).IDE was 1:

IDF	Meaning
0	Input denormal exception has not occurred.
1	Input denormal exception has occurred.

Input Denormal exceptions can occur only when [FPSCR](#).FZ is 1.

This bit must be cleared to 0 by the exception-handling routine.

When the value of FPEXC.TFV is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions this bit is RAZ/WI.

#### Bits [6:5]

Reserved, RES0.

#### IXF, bit [4]

Inexact trapped exception bit. Valid only when the value of FPEXC.TFV is 1. When valid, it indicates whether an Inexact exception occurred while [FPSCR](#).IXE was 1:

IXF	Meaning
0	Inexact exception has not occurred.
1	Inexact exception has occurred.

This bit must be cleared to 0 by the exception-handling routine.

When the value of FPEXC.TFV is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions this bit is RAZ/WI.

#### UFF, bit [3]

Underflow trapped exception bit. Valid only when the value of FPEXC.TFV is 1. When valid, it indicates whether an Underflow exception occurred while [FPSCR](#).UFE was 1:

UFF	Meaning
0	Underflow exception has not occurred.
1	Underflow exception has occurred.

Underflow trapped exceptions can occur only when [FPSCR](#).FZ is 0.

This bit must be cleared to 0 by the exception-handling routine.

When the value of FPEXC.TFV is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions this bit is RAZ/WI.

**OFF, bit [2]**

Overflow trapped exception bit. Valid only when the value of FPEXC.TFV is 1. When valid, it indicates whether an Overflow exception occurred while [FPSCR.OFE](#) was 1:

OFF	Meaning
0	Overflow exception has not occurred.
1	Overflow exception has occurred.

This bit must be cleared to 0 by the exception-handling routine.

When the value of FPEXC.TFV is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions this bit is RAZ/WI.

**DZF, bit [1]**

Divide by Zero trapped exception bit. Valid only when the value of FPEXC.TFV is 1. When valid, it indicates whether a Divide by Zero exception occurred while [FPSCR.DZE](#) was 1:

DZF	Meaning
0	Divide by Zero exception has not occurred.
1	Divide by Zero exception has occurred.

This bit must be cleared to 0 by the exception-handling routine.

When the value of FPEXC.TFV is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions this bit is RAZ/WI.

**IOF, bit [0]**

Invalid Operation trapped exception bit. Valid only when the value of FPEXC.TFV is 1. When valid, it indicates whether an Invalid Operation exception occurred while [FPSCR.IOE](#) was 1:

IOF	Meaning
0	Invalid Operation exception has not occurred.
1	Invalid Operation exception has occurred.

This bit must be cleared to 0 by the exception-handling routine.

When the value of FPEXC.TFV is 0 and this bit is RW, this bit is invalid and UNKNOWN.

On an implementation that does not support the trapping of floating-point exceptions this bit is RAZ/WI.

**Accessing the FPEXC32\_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
FPEXC32_EL2	11	100	0101	0011	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 [CPTR\\_EL2](#).TFP==1, Non-secure accesses to this register from EL2 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 [CPTR\\_EL2](#).FPEN==00, Non-secure accesses to this register from EL2 are trapped to EL2.
- If [CPTR\\_EL2](#).FPEN==10, Non-secure accesses to this register from EL2 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 [CPTR\\_EL2](#).FPEN==00, Non-secure accesses to this register from EL2 are trapped to EL2.
- If [CPTR\\_EL2](#).FPEN==10, Non-secure accesses to this register from EL2 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [CPTR\\_EL3](#).TFP==1, accesses to this register from EL2 and EL3 are trapped to EL3.



# FPSR, Floating-point Status Register

The FPSR characteristics are:

## Purpose

Provides floating-point system status information.

This register is part of:

- The Special-purpose registers functional group.
- The Floating-point registers functional group.

## Configuration

The named fields in this register map to the equivalent fields in the AArch32 [FPSCR](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

FPSR is a 32-bit register.

## Field descriptions

The FPSR 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
N	Z	C	V	QC	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IDC	0	0	IXC	UFC	OFC	DZC	IOC

### N, bit [31]

Negative condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.N flag instead.

### Z, bit [30]

Zero condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.Z flag instead.

### C, bit [29]

Carry condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.C flag instead.

### V, bit [28]

Overflow condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.V flag instead.

### QC, bit [27]

Cumulative saturation bit, Advanced SIMD only. This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated since 0 was last written to this bit.

### Bits [26:8]

Reserved, RES0.

**IDC, bit [7]**

Input Denormal cumulative floating-point exception bit. This bit is set to 1 to indicate that the Input Denormal floating-point exception has occurred since 0 was last written to this bit.

How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the [FPCR.IDE](#) bit. This bit is only set to 1 to indicate a floating-point exception if [FPCR.IDE](#) is 0, or if trapping software sets it.

**Bits [6:5]**

Reserved, RES0.

**IXC, bit [4]**

Inexact cumulative floating-point exception bit. This bit is set to 1 to indicate that the Inexact exception floating-point has occurred since 0 was last written to this bit.

How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the [FPCR.IXE](#) bit. This bit is only set to 1 to indicate a floating-point exception if [FPCR.IXE](#) is 0, or if trapping software sets it.

**UFC, bit [3]**

Underflow cumulative floating-point exception bit. This bit is set to 1 to indicate that the Underflow floating-point exception has occurred since 0 was last written to this bit.

How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the [FPCR.UFE](#) bit. This bit is only set to 1 to indicate a floating-point exception if [FPCR.UFE](#) is 0, or if trapping software sets it.

**OFC, bit [2]**

Overflow cumulative floating-point exception bit. This bit is set to 1 to indicate that the Overflow floating-point exception has occurred since 0 was last written to this bit.

How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the [FPCR.OFE](#) bit. This bit is only set to 1 to indicate a floating-point exception if [FPCR.OFE](#) is 0, or if trapping software sets it.

**DZC, bit [1]**

Divide by Zero cumulative floating-point exception bit. This bit is set to 1 to indicate that the Divide by Zero floating-point exception has occurred since 0 was last written to this bit.

How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the [FPCR.DZE](#) bit. This bit is only set to 1 to indicate a floating-point exception if [FPCR.DZE](#) is 0, or if trapping software sets it.

**IOC, bit [0]**

Invalid Operation cumulative floating-point exception bit. This bit is set to 1 to indicate that the Invalid Operation floating-point exception has occurred since 0 was last written to this bit.

How scalar and Advanced SIMD floating-point instructions update this bit depends on the value of the [FPCR.IOE](#) bit. This bit is only set to 1 to indicate a floating-point exception if [FPCR.IOE](#) is 0, or if trapping software sets it.

## Accessing the FPSR

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
FPSR	11	011	0100	0100	001

## 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.

When `HCR_EL2.E2H==0` :

- If [CPACR\\_EL1.FPEN==00](#), accesses to this register from EL0 and EL1 are trapped to EL1.
- If [CPACR\\_EL1.FPEN==01](#), accesses to this register from EL0 are trapped to EL1.
- If [CPACR\\_EL1.FPEN==10](#), accesses to this register from EL0 and EL1 are trapped to EL1.

When EL2 is implemented and is using AArch64 and `SCR_EL3.NS==1 && HCR_EL2.E2H==0` :

- If [CPTR\\_EL2.TFP==1](#), Non-secure accesses to this register from EL0, EL1, and EL2 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 [CPACR\\_EL1.FPEN==00](#), Non-secure accesses to this register from EL0 and EL1 are trapped to EL1.
- If [CPACR\\_EL1.FPEN==01](#), Non-secure accesses to this register from EL0 are trapped to EL1.
- If [CPACR\\_EL1.FPEN==10](#), Non-secure accesses to this register from EL0 and EL1 are trapped to EL1.
- If [CPTR\\_EL2.FPEN==00](#), Non-secure accesses to this register from EL0, EL1, and EL2 are trapped to EL2.
- If [CPTR\\_EL2.FPEN==10](#), Non-secure accesses to this register from EL0, EL1, and EL2 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 [CPTR\\_EL2.FPEN==00](#), Non-secure accesses to this register from EL0 and EL2 are trapped to EL2.
- If [CPTR\\_EL2.FPEN==01](#), Non-secure accesses to this register from EL0 are trapped to EL2.
- If [CPTR\\_EL2.FPEN==10](#), Non-secure accesses to this register from EL0 and EL2 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [CPTR\\_EL3.TFP==1](#), accesses to this register from EL0, EL1, EL2, and EL3 are trapped to EL3.

# HACR\_EL2, Hypervisor Auxiliary Control Register

The HACR\_EL2 characteristics are:

## Purpose

Controls trapping to EL2 of IMPLEMENTATION DEFINED aspects of Non-secure EL1 or EL0 operation.

**Note**

ARM recommends the values in this register do not cause unnecessary traps to EL2 when [HCR\\_EL2](#).{E2H, TGE} = {1, 1}.

This register is part of:

- The Virtualization registers functional group.
- The IMPLEMENTATION DEFINED functional group.

## Configuration

AArch64 System register HACR\_EL2 is architecturally mapped to AArch32 System register [HACR](#).

If EL2 is not implemented, this register is RES0 from EL3.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HACR\_EL2 is a 32-bit register.

## Field descriptions

The HACR\_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
IMPLEMENTATION DEFINED																															

**IMPLEMENTATION DEFINED, bits [31:0]**

IMPLEMENTATION DEFINED.

## Accessing the HACR\_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
HACR_EL2	11	100	0001	0001	111

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

## Purpose

This register is part of the Virtualization registers functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

## Field descriptions

63	62	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	0	0	0	0	0	0	0	MIOCNCE	T
RW	TRVM	HCD	TDZ	TGETVM	TTLB	TPU	TPCP	TSW	TACR	TIDCPT	TSC	TID3	TID2	TID1	TID0	TWE	TWI	DCBSU	FB	VSEVI	VF	A					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6		

MIOCNE	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.

Page 1066

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 ARMv8.2 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 ARMv8.2 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 <a href="#">DC ZVA</a> instruction at Non-secure EL1, or at Non-secure EL0 when the instruction is not UNDEFINED at EL0, is trapped to EL2. Reading the <a href="#">DCZID_EL0</a> returns a value that indicates that <a href="#">DC ZVA</a> 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 <a href="#">SCR_EL3.NS</a> is 0, this control has no effect on execution at EL0.</p> <p>When the value of <a href="#">SCR_EL3.NS</a> is 1, in all cases:</p> <ul style="list-style-type: none"> <li>• All exceptions that would be routed to EL1 are routed to EL2.</li> <li>• The <a href="#">SCTLR_EL1.M</a> field, or the <a href="#">SCTLR.M</a> field if EL1 is using AArch32, is treated as being 0 for all purposes other than returning the result of a direct read of <a href="#">SCTLR_EL1</a> or <a href="#">SCTLR</a>.</li> <li>• All virtual interrupts are disabled.</li> <li>• Any IMPLEMENTATION DEFINED mechanisms for signaling virtual interrupts are disabled.</li> <li>• An exception return to EL1 is treated as an illegal exception return.</li> </ul> <p>When the value of <a href="#">SCR_EL3.NS</a> is 1 and the value of <a href="#">HCR_EL2.E2H</a> is 0, additionally:</p> <ul style="list-style-type: none"> <li>• The <a href="#">HCR_EL2.{FMO, IMO, AMO}</a> fields are treated as being 1 for all purposes other than a direct read or write access of <a href="#">HCR_EL2</a>.</li> <li>• The <a href="#">MDCR_EL2.{TDRA, TDOSA, TDA, TDE}</a> fields are treated as being 1 for all purposes other than returning the result of a direct read of <a href="#">MDCR_EL2</a>.</li> </ul> <p>For information on the behavior of this bit when <a href="#">E2H</a> is 1, see Behavior of <a href="#">HCR_EL2.E2H</a>.</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:

Non-secure EL1 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](#).

Non-secure EL1 using AArch32: [TLBIALLIS](#), [TLBIMVAIS](#), [TLBIASIDIS](#), [TLBIMVAIS](#), [TLBIMVALIS](#), [TLBIMVAALIS](#), [ITLBIALL](#), [ITLBMVA](#), [ITLBIASID](#), [DTLBIALL](#), [DTLBMVA](#), [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:

Non-secure EL0 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.

Non-secure EL1 using AArch64: [IC IVAU](#), [IC IALLU](#), [IC IALLUIS](#), [DC CVAU](#).

Non-secure EL1 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.

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 ARMv8.2:

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:

Non-secure EL0 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.

Non-secure EL1 using AArch64: [DC IVAC](#), [DC CIVAC](#), [DC CVAC](#), [DC CVAP](#).

Non-secure EL1 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.

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.

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.

#### Note

In previous versions of the architecture this bit was named TPC. From ARMv8.2 this bit is named TPCP.

#### 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:

Non-secure EL0 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.

Non-secure EL1 using AArch64: [DC IVAC](#), [DC CIVAC](#), [DC CVAC](#).

Non-secure EL1 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.

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.

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:

Non-secure EL1 using AArch64: [DC ISW](#), [DC CSW](#), [DC CISW](#).

Non-secure EL1 using AArch32: [DCISW](#), [DCCSW](#), [DCCISW](#).

#### 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	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 <a href="#">SCR_EL3.SMD</a> .

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, 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](#), [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](#), [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 <a href="#">SCTLR.nTWE</a> or <a href="#">SCTLR_EL1.nTWE</a> .

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.

<b>TWI</b>	<b>Meaning</b>
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 <a href="#">SCTLR.nTWI</a> or <a href="#">SCTLR_EL1.nTWI</a> .

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]**

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:

<b>BSU</b>	<b>Meaning</b>
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](#), [ICALLU](#), [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 <a href="#">SCR_EL3.EA</a> bit. Virtual SError interrupts are disabled.
1	When executing at any Exception level in Non-secure state: <ul style="list-style-type: none"> <li>Physical SError interrupts are taken to EL2 unless they are routed to EL3.</li> <li>If HCR_EL2.TGE==0 then Virtual SError interrupts are enabled in the Non-secure state.</li> </ul>

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 <a href="#">SCR_EL3.IRQ</a> bit. Virtual IRQ interrupts are disabled.
1	When executing at any Exception level in Non-secure state: <ul style="list-style-type: none"> <li>• Physical IRQ interrupts are taken to EL2 unless they are routed to EL3.</li> <li>• If HCR_EL2.TGE==0 then Virtual IRQ interrupts are enabled in Non-secure state.</li> </ul>

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 <a href="#">SCR_EL3.FIQ</a> bit. Virtual FIQ interrupts are disabled.
1	When executing at any Exception level in Non-secure state: <ul style="list-style-type: none"> <li>• Physical FIQ interrupts are taken to EL2 unless they are routed to EL3.</li> <li>• If HCR_EL2.TGE==0 then Virtual FIQ interrupts are enabled in Non-secure state.</li> </ul>

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: [DC ISW](#) performs the same invalidation as a [DC CISW](#) 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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 ARMv8.2:

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. 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.

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.

# HSTR\_EL2, Hypervisor System Trap Register

The HSTR\_EL2 characteristics are:

## Purpose

Controls trapping to Hyp mode of Non-secure accesses, at EL1 or lower in AArch32, to the System register in the coproc == 1111 encoding space, by the CRn value used to access the register using MCR or MRC instruction. When the register is accessible using an MCRR or MRRC instruction, this is the CRm value used to access the register.

This register is part of the Virtualization registers functional group.

## Configuration

AArch64 System register HSTR\_EL2 is architecturally mapped to AArch32 System register [HSTR](#).

If EL2 is not implemented, this register is RES0 from EL3.

If no Exception level can use AArch32, then this register is RES0

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

HSTR\_EL2 is a 32-bit register.

## Field descriptions

The HSTR\_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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	T15	T14	T13	T12	T11	T10	T9	T8	T7	T6	T5	T4	T3	T2	T1	T0

### Bits [31:16]

Reserved, RES0.

### T<n>, bit [n], for n = 0 to 15

Fields T14 and T4 are RES0.

The remaining fields control whether Non-secure EL0 and EL1 accesses, using MCR, MRC, MCRR, and MRRC instructions, to the System registers in the coproc == 1111 encoding space are trapped to Hyp mode:

T<n>	Meaning
0	This control has no effect on Non-secure EL0 or EL1 accesses to System registers.
1	Any Non-secure EL1 MCR, MRC access with coproc == 1111 and CRn == <n> is trapped to Hyp mode if the access is not UNDEFINED when the value of this field is 0. Any Non-secure EL1 MCRR, MRRC access with coproc == 1111 and CRm == <n> is trapped to Hyp mode if the access is not UNDEFINED when the value of this field is 0.

For example, when HSTR\_EL2.T7 is 1:

- Any 32-bit access from a Non-secure EL1 mode, using an MCR or MRC instruction with coproc set to 1111 and <CRn> set to c7, and that is not UNDEFINED when HSTR\_EL2.T7 is 0, is trapped to Hyp mode.
- Any 64-bit access from a Non-secure EL1 mode, using an MCRR or MRRC instruction with coproc set to 1111 and <CRm> set to c7, and that is not UNDEFINED when HSTR\_EL2.T7 is 0, is trapped to Hyp mode.

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 HSTR\_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
HSTR_EL2	11	100	0001	0001	011

## 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.



# ICC\_AP0R<n>\_EL1, Interrupt Controller Active Priorities Group 0 Registers, n = 0 - 3

The ICC\_AP0R<n>\_EL1 characteristics are:

## Purpose

Provides information about Group 0 active priorities.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_AP0R<n>\_EL1 is architecturally mapped to AArch32 System register [ICC\\_AP0R<n>](#).

## Attributes

ICC\_AP0R<n>\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_AP0R<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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

When this register has an architecturally-defined reset value, this field resets to 0.

The contents of these registers are IMPLEMENTATION DEFINED with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

## Accessing the ICC\_AP0R<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>	op1	CRn	CRm	op2
ICC_AP0R<n>_EL1	000	1100	1000	1:n<1:0>

When HCR\_EL2.FMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_AP0R<n>\\_EL1](#).

## 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	-	<a href="#">ICV_AP0R&lt;n&gt;_EL1</a>	RW	RW

This table applies to all instructions that can access this register.

The ICC\_AP0R<n>\_EL1 registers are only accessible at Non-secure EL1 when HCR\_EL2.FMO is set to 0.

### Note

When HCR\_EL2.FMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC\_AP0R<n>\_EL1 results in an access to [ICV\\_AP0R<n>\\_EL1](#).

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 0 active priorities) might result in UNPREDICTABLE behavior of the interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICC\_AP0R1\_EL1 is only implemented in implementations that support 6 or more bits of priority. ICC\_AP0R2\_EL1 and ICC\_AP0R3\_EL1 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- ICC\_AP0R<n>\_EL1.
- Secure [ICC\\_APIR<n>\\_EL1](#).
- Non-secure [ICC\\_APIR<n>\\_EL1](#).

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- 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 AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).FIQ==1, Secure accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).FIQ==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](#).FIQ==1, and [HCR\\_EL2](#).FMO==0, Non-secure accesses to this register from EL1 are trapped to EL3.



# ICC\_AP1R<n>\_EL1, Interrupt Controller Active Priorities Group 1 Registers, n = 0 - 3

The ICC\_AP1R<n>\_EL1 characteristics are:

## Purpose

Provides information about Group 1 active priorities.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_AP1R<n>\_EL1 (S) is architecturally mapped to AArch32 System register [ICC\\_AP1R<n> \(S\)](#).

AArch64 System register ICC\_AP1R<n>\_EL1 (NS) is architecturally mapped to AArch32 System register [ICC\\_AP1R<n> \(NS\)](#).

## Attributes

ICC\_AP1R<n>\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_AP1R<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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

When this register has an architecturally-defined reset value, this field resets to 0.

The contents of these registers are IMPLEMENTATION DEFINED with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

## Accessing the ICC\_AP1R<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>	op1	CRn	CRm	op2
ICC_AP1R<n>_EL1	000	1100	1001	0:n<1:0>

When HCR\_EL2.IMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_AP1R<n>\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility				Instance
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3	
x	x	x	0	-	RW	n/a	RW	ICC_AP1R<n>_EL1_s
x	x	1	1	-	n/a	RW	RW	ICC_AP1R<n>_EL1_ns
x	0	0	1	-	RW	RW	RW	ICC_AP1R<n>_EL1_ns
x	1	0	1	-	<a href="#">ICV_AP1R&lt;n&gt;_EL1</a>	RW	RW	ICC_AP1R<n>_EL1_ns

This table applies to all instructions that can access this register.

The ICC\_AP1R<n>\_EL1 registers are only accessible at Non-secure EL1 when HCR\_EL2.IMO is set to 0.

### Note

When HCR\_EL2.IMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC\_AP1R<n>\_EL1 results in an access to [ICV\\_AP1R<n>\\_EL1](#).

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 1 active priorities) might result in UNPREDICTABLE behavior of the interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICC\_AP1R1\_EL1 is only implemented in implementations that support 6 or more bits of priority. ICC\_AP1R2\_EL1 and ICC\_AP1R3\_EL1 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- [ICC\\_AP0R<n>\\_EL1](#).
- Secure ICC\_AP1R<n>\_EL1.
- Non-secure ICC\_AP1R<n>\_EL1.

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- 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 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\\_EL2](#).IMO==0, Non-secure accesses to this register from EL1 are trapped to EL3.



# ICC\_ASGI1R\_EL1, Interrupt Controller Alias Software Generated Interrupt Group 1 Register

The ICC\_ASGI1R\_EL1 characteristics are:

## Purpose

Generates Group 1 SGIs for the Security state that is not the current Security state.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_ASGI1R\_EL1 performs the same function as AArch32 System register [ICC\\_ASGI1R](#).

Under certain conditions a write to ICC\_ASGI1R\_EL1 can generate Group 0 interrupts, see Forwarding an SGI to a target PE.

## Attributes

ICC\_ASGI1R\_EL1 is a 64-bit register.

## Field descriptions

The ICC\_ASGI1R\_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	Aff3								0	0	0	0	0	0	0	IRM	Aff2									
0	0	0	0	INTID				Aff1								TargetList																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

### Bits [63:56]

Reserved, RES0.

### Aff3, bits [55:48]

The affinity 3 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

### Bits [47:41]

Reserved, RES0.

### IRM, bit [40]

Interrupt Routing Mode. Determines how the generated interrupts are distributed to PEs. Possible values are:

IRM	Meaning
0	Interrupts routed to the PEs specified by Aff3.Aff2.Aff1.<target list>.
1	Interrupts routed to all PEs in the system, excluding "self".

**Aff2, bits [39:32]**

The affinity 2 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

**Bits [31:28]**

Reserved, RES0.

**INTID, bits [27:24]**

The INTID of the SGI.

**Aff1, bits [23:16]**

The affinity 1 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

**TargetList, bits [15:0]**

Target List. The set of PEs for which SGI interrupts will be generated. Each bit corresponds to the PE within a cluster with an Affinity 0 value equal to the bit number.

If a bit is 1 and the bit does not correspond to a valid target PE, the bit must be ignored by the Distributor. It is IMPLEMENTATION DEFINED whether, in such cases, a Distributor can signal a system error.

**Note**

This restricts a system to sending targeted SGIs to PEs with an affinity 0 number that is less than 16.

If SRE is set only for Secure EL3, software executing at EL3 might use the System register interface to generate SGIs. Therefore, the Distributor must always be able to receive and acknowledge Generate SGI packets received from CPU interface regardless of the ARE settings for a Security state. However, the Distributor might discard such packets.

If the IRM bit is 1, this field is RES0.

## Accessing the ICC\_ASGI1R\_EL1

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>	op1	CRn	CRm	op2
ICC_ASGI1R_EL1	000	1100	1011	110

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	WO	n/a	WO
0	1	-	WO	WO	WO
1	1	-	n/a	WO	WO



This table applies to all instructions that can access this register.

This register allows software executing in a Secure state to generate Non-secure Group 1 SGIs. It will also allow software executing in a Non-secure state to generate Secure Group 1 SGIs, if permitted by the settings of [GICR\\_NSACR](#) in the Redistributor corresponding to the target PE.

When [GICD\\_CTLR.DS](#)==0, Non-secure writes do not generate an interrupt for a target PE if not permitted by the [GICR\\_NSACR](#) register associated with the target PE. For more information see Use of control registers for SGI forwarding.

---

#### Note

Accesses at EL3 are treated as Secure regardless of the value of SCR\_EL3.NS.

---

## 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 [ICC\\_SRE\\_EL1.SRE](#)==0, write accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2.SRE](#)==0, write accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3.SRE](#)==0, write accesses to this register from EL3 are trapped to EL3.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HCR\\_EL2.FMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR\\_EL2.IMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HCR\\_EL2.FMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR\\_EL2.IMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2.TC](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3.FIQ](#)==1, and [SCR\\_EL3.IRQ](#)==1, Secure write accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3.FIQ](#)==1, and [SCR\\_EL3.IRQ](#)==1, write 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, [SCR\\_EL3.IRQ](#)==1, [HCR\\_EL2.IMO](#)==0, and [HCR\\_EL2.FMO](#)==0, Non-secure write accesses to this register from EL1 are trapped to EL3.

# ICC\_BPR0\_EL1, Interrupt Controller Binary Point Register 0

The ICC\_BPR0\_EL1 characteristics are:

## Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 0 interrupt preemption.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_BPR0\_EL1 is architecturally mapped to AArch32 System register [ICC\\_BPR0](#).

Virtual accesses to this register update [ICH\\_VMCR\\_EL2](#).VBPR0.

## Attributes

ICC\_BPR0\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_BPR0\_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	0	0	BinaryPoint

### Bits [31:3]

Reserved, RES0.

### BinaryPoint, bits [2:0]

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. This is done as follows:

Binary point value	Group priority field	Subpriority field	Field with binary point
0	[7:1]	[0]	ggggggg.s
1	[7:2]	[1:0]	ggggggg.ss
2	[7:3]	[2:0]	ggggggg.sss
3	[7:4]	[3:0]	ggggg.ssss
4	[7:5]	[4:0]	ggg.sssss
5	[7:6]	[5:0]	gg.ssssss
6	[7]	[6:0]	g.sssssss
7	No preemption	[7:0]	.ssssssss

## Accessing the ICC\_BPR0\_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 &lt;systemreg&gt;, &lt;Xt&gt;

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op1	CRn	CRm	op2
ICC_BPR0_EL1	000	1100	1000	011

When HCR\_EL2.FMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_BPR0\\_EL1](#).

## 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	-	<a href="#">ICV_BPR0_EL1</a>	RW	RW

This table applies to all instructions that can access this register.

ICC\_BPR0\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.FMO is set to 0.

### Note

When HCR\_EL2.FMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC\_BPR0\_EL1 results in an access to [ICV\\_BPR0\\_EL1](#).

The minimum binary point value is derived from the number of implemented priority bits. The number of priority bits is IMPLEMENTATION DEFINED, and reported by [ICC\\_CTLR\\_EL1.PRIBits](#) and [ICC\\_CTLR\\_EL3.PRIBits](#).

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value. On a reset, the binary point field is UNKNOWN.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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 [ICC\\_SRE\\_EL1.SRE](#)==0, accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2.SRE](#)==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3.SRE](#)==0, accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- 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 AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3.FIQ](#)==1, Secure accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3.FIQ](#)==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.FIQ](#)==1, and [HCR\\_EL2.FMO](#)==0, Non-secure accesses to this register from EL1 are trapped to EL3.



# ICC\_BPR1\_EL1, Interrupt Controller Binary Point Register 1

The ICC\_BPR1\_EL1 characteristics are:

## Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_BPR1\_EL1 (S) is architecturally mapped to AArch32 System register [ICC\\_BPR1 \(S\)](#).

AArch64 System register ICC\_BPR1\_EL1 (NS) is architecturally mapped to AArch32 System register [ICC\\_BPR1 \(NS\)](#).

Virtual accesses to this register update [ICH\\_VMCR\\_EL2.VBPR1](#).

## Attributes

ICC\_BPR1\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_BPR1\_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	0	0	BinaryPoint

### Bits [31:3]

Reserved, RES0.

### BinaryPoint, bits [2:0]

If the GIC is configured to use separate binary point fields for Group 0 and Group 1 interrupts, the value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. For more information about priorities, see Priority grouping.

The minimum value of the Non-secure copy of this register is the minimum value of [ICC\\_BPR0\\_EL1](#) + 1. The minimum value of the Secure copy of this register is the minimum value of [ICC\\_BPR0\\_EL1](#).

If EL3 is implemented and [ICC\\_CTLR\\_EL3.CBPR\\_EL1S](#) is 1:

- Writing to this register at Secure EL1 modifies [ICC\\_BPR0\\_EL1](#).
- Reading this register at Secure EL1 returns the value of [ICC\\_BPR0\\_EL1](#).

If EL3 is implemented and [ICC\\_CTLR\\_EL3.CBPR\\_EL1NS](#) is 1, Non-secure accesses to this register at EL1 or EL2 behave as follows, depending on the values of [HCR\\_EL2.IMO](#) and [SCR\\_EL3.IRQ](#):

HCR_EL2.IMO	SCR_EL3.IRQ	Behavior
0	0	Non-secure EL1 and EL2 reads return <a href="#">ICC_BPR0_EL1</a> + 1 saturated to 0b111. Non-secure EL1 and EL2 writes are ignored.
0	1	Non-secure EL1 and EL2 accesses trap to EL3.
1	0	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 reads return <a href="#">ICC_BPR0_EL1</a> + 1 saturated to 0b111. Non-secure EL2 writes are ignored.
1	1	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 accesses trap to EL3.

If EL3 is not implemented and [ICC\\_CTLR\\_EL1](#).CBPR is 1, Non-secure accesses to this register at EL1 or EL2 behave as follows, depending on the values of HCR\_EL2.IMO:

HCR_EL2.IMO	Behavior
0	Non-secure EL1 and EL2 reads return <a href="#">ICC_BPR0_EL1</a> + 1 saturated to 0b111. Non-secure EL1 and EL2 writes are ignored.
1	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 reads return <a href="#">ICC_BPR0_EL1</a> + 1 saturated to 0b111. Non-secure EL2 writes are ignored.

## Accessing the ICC\_BPR1\_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>	op1	CRn	CRm	op2
ICC_BPR1_EL1	000	1100	1100	011

When HCR\_EL2.IMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_BPR1\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility				Instance
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3	
x	x	x	0	-	RW	n/a	RW	ICC_BPR1_EL1_s
x	x	1	1	-	n/a	RW	RW	ICC_BPR1_EL1_ns
x	0	0	1	-	RW	RW	RW	ICC_BPR1_EL1_ns
x	1	0	1	-	<a href="#">ICV_BPR1_EL1</a>	RW	RW	ICC_BPR1_EL1_ns

This table applies to all instructions that can access this register.

ICC\_BPR1\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.IMO is set to 0.

### Note

When HCR\_EL2.IMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC\_BPR1\_EL1 results in an access to [ICV\\_BPR1\\_EL1](#).

On a reset, the binary point field is UNKNOWN.

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value.

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, accesses to this register from EL1 generate an Undefined exception that is taken to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- 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 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\\_EL2](#).IMO==0, Non-secure accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICC\_CTLR\_EL1, Interrupt Controller Control Register (EL1)

The ICC\_CTLR\_EL1 characteristics are:

## Purpose

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_CTLR\_EL1 (S) is architecturally mapped to AArch32 System register [ICC\\_CTLR \(S\)](#).

AArch64 System register ICC\_CTLR\_EL1 (NS) is architecturally mapped to AArch32 System register [ICC\\_CTLR \(NS\)](#).

## Attributes

ICC\_CTLR\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_CTLR\_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	<a href="#">A3V</a>	<a href="#">SEIS</a>	<a href="#">IDbits</a>	<a href="#">PRIbits</a>	0	<a href="#">PMHE</a>	0	0	0	0	<a href="#">EOImode</a>	<a href="#">CBPR</a>				

### Bits [31:16]

Reserved, RES0.

### A3V, bit [15]

Affinity 3 Valid. Read-only and writes are ignored. Possible values are:

A3V	Meaning
0	The CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers.
1	The CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

If EL3 is implemented, this bit is an alias of [ICC\\_CTLR\\_EL3](#).A3V.

### SEIS, bit [14]

SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs:

SEIS	Meaning
0	The CPU interface logic does not support local generation of SEIs.
1	The CPU interface logic supports local generation of SEIs.

If EL3 is implemented, this bit is an alias of [ICC\\_CTLR\\_EL3](#).SEIS.



**IDbits, bits [13:11]**

Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported:

IDbits	Meaning
000	16 bits.
001	24 bits.

All other values are reserved.

If EL3 is implemented, this field is an alias of [ICC\\_CTLR\\_EL3](#).IDbits.

**PRbits, bits [10:8]**

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).

An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).

**Note**

This field always returns the number of priority bits implemented, regardless of the Security state of the access or the value of [GICD\\_CTLR](#).DS.

For physical accesses, this field determines the minimum value of [ICC\\_BPR0\\_EL1](#).

If EL3 is implemented, physical accesses return the value from [ICC\\_CTLR\\_EL3](#).PRbits.

If EL3 is not implemented, physical accesses return the value from this field.

**Bit [7]**

Reserved, RES0.

**PMHE, bit [6]**

Priority Mask Hint Enable. Controls whether the priority mask register is used as a hint for interrupt distribution:

PMHE	Meaning
0	Disables use of <a href="#">ICC_PMR_EL1</a> as a hint for interrupt distribution.
1	Enables use of <a href="#">ICC_PMR_EL1</a> as a hint for interrupt distribution.

If EL3 is implemented, this bit is an alias of [ICC\\_CTLR\\_EL3](#).PMHE. Whether this bit can be written as part of an access to this register depends on the value of [GICD\\_CTLR](#).DS:

- If [GICD\\_CTLR](#).DS == 0, this bit is read-only.
- If [GICD\\_CTLR](#).DS == 1, this bit is read/write.

If EL3 is not implemented, it is IMPLEMENTATION DEFINED whether this bit is read-only or read-write:

- If this bit is read-only, an implementation can choose to make this field RAZ/WI or RAO/WI.
- If this bit is read/write, it resets to zero.

**Bits [5:2]**

Reserved, RES0.

**EOImode, bit [1]**

EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt:

EOImode	Meaning
0	<a href="#">ICC_EOIR0_EL1</a> and <a href="#">ICC_EOIR1_EL1</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">ICC_DIR_EL1</a> are UNPREDICTABLE.
1	<a href="#">ICC_EOIR0_EL1</a> and <a href="#">ICC_EOIR1_EL1</a> provide priority drop functionality only. <a href="#">ICC_DIR_EL1</a> provides interrupt deactivation functionality.

The Secure [ICC\\_CTLR\\_EL1](#).EOImode is an alias of [ICC\\_CTLR\\_EL3](#).EOImode\_EL1S.

The Non-secure [ICC\\_CTLR\\_EL1](#).EOImode is an alias of [ICC\\_CTLR\\_EL3](#).EOImode\_EL1NS.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### CBPR, bit [0]

Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupts:

CBPR	Meaning
0	<a href="#">ICC_BPR0_EL1</a> determines the preemption group for Group 0 interrupts only. <a href="#">ICC_BPR1_EL1</a> determines the preemption group for Group 1 interrupts.
1	<a href="#">ICC_BPR0_EL1</a> determines the preemption group for both Group 0 and Group 1 interrupts.

If EL3 is implemented:

- This bit is an alias of [ICC\\_CTLR\\_EL3](#).CBPR\_EL1 {S,NS} where S or NS corresponds to the current Security state.
- If [GICD\\_CTLR](#).DS == 0, this bit is read-only.
- If [GICD\\_CTLR](#).DS == 1, this bit is read/write.

If EL3 is not implemented, this bit is read/write.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the ICC\_CTLR\_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>	op1	CRn	CRm	op2
ICC_CTLR_EL1	000	1100	1100	100

When HCR\_EL2.{FMO, IMO} != {0, 0}, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_CTLR\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility				Instance
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3	
x	x	1	1	-	n/a	RW	RW	ICC_CTLR_EL1_ns
x	1	0	1	-	<a href="#">ICV_CTLR_EL1</a>	RW	RW	ICC_CTLR_EL1_ns
1	x	0	1	-	<a href="#">ICV_CTLR_EL1</a>	RW	RW	ICC_CTLR_EL1_ns
0	0	0	1	-	RW	RW	RW	ICC_CTLR_EL1_ns
x	x	x	0	-	RW	n/a	RW	ICC_CTLR_EL1_s

This table applies to all instructions that can access this register.

ICC\_CTLR\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.{FMO, IMO} == {0, 0}.

#### Note

When HCR\_EL2.{FMO, IMO} != {0, 0}, at Non-secure EL1, the instruction encoding used to access ICC\_CTLR\_EL1 results in an access to [ICV\\_CTLR\\_EL1](#).

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).FIQ==1, and [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](#).FIQ==1, and [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](#).FIQ==1, [SCR\\_EL3](#).IRQ==1, [HCR\\_EL2](#).IMO==0, and [HCR\\_EL2](#).FMO==0, Non-secure accesses to this register from EL1 are trapped to EL3.

# ICC\_CTLR\_EL3, Interrupt Controller Control Register (EL3)

The ICC\_CTLR\_EL3 characteristics are:

## Purpose

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

This register is part of:

- The GIC system registers functional group.
- The Security registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_CTLR\_EL3 can be mapped to AArch32 System register [ICC\\_MCTLR](#), but this is not architecturally mandated.

## Attributes

ICC\_CTLR\_EL3 is a 32-bit register.

## Field descriptions

The ICC\_CTLR\_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	0	0	0	0	0	0	0	0	0	0	0	0	0	nDS	0	A3V	SEIS	IDbits	PRIBits	0	PMHE	RMEOImode_EL1	NS	EOImode_EL1	SE	EOImode_EL3	CBPR_E			

### Bits [31:18]

Reserved, RES0.

### nDS, bit [17]

Disable Security not supported. Read-only and writes are ignored. Possible values are:

nDS	Meaning
0	The CPU interface logic supports disabling of security.
1	The CPU interface logic does not support disabling of security, and requires that security is not disabled.

### Bit [16]

Reserved, RES0.

### A3V, bit [15]

Affinity 3 Valid. Read-only and writes are ignored. Possible values are:

A3V	Meaning
0	The CPU interface logic does not support non-zero values of the Aff3 field in SGI generation System registers.
1	The CPU interface logic supports non-zero values of the Aff3 field in SGI generation System registers.

If EL3 is present, [ICC\\_CTLR\\_EL1](#).AV3 is an alias of ICC\_CTLR\_EL3.A3V

**SEIS, bit [14]**

SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports generation of SEIs:

SEIS	Meaning
0	The CPU interface logic does not support generation of SEIs.
1	The CPU interface logic supports generation of SEIs.

If EL3 is present, [ICC\\_CTLR\\_EL1](#).SEIS is an alias of ICC\_CTLR\_EL3.SEIS

**IDbits, bits [13:11]**

Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported:

IDbits	Meaning
000	16 bits.
001	24 bits.

All other values are reserved.

If EL3 is present, [ICC\\_CTLR\\_EL1](#).IDbits is an alias of ICC\_CTLR\_EL3.IDbits

**PRibits, bits [10:8]**

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).

An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).

**Note**

This field always returns the number of priority bits implemented, regardless of the value of SCR\_EL3.NS or the value of [GICD\\_CTLR](#).DS.

The division between group priority and subpriority is defined in the binary point registers [ICC\\_BPR0\\_EL1](#) and [ICC\\_BPR1\\_EL1](#).

This field determines the minimum value of ICC\_BPR0\_EL1.

**Bit [7]**

Reserved, RES0.

**PMHE, bit [6]**

Priority Mask Hint Enable.

PMHE	Meaning
0	Disables use of the priority mask register as a hint for interrupt distribution.
1	Enables use of the priority mask register as a hint for interrupt distribution.

Software must write [ICC\\_PMR\\_EL1](#) to 0xFF before clearing this field to 0.

- An implementation might choose to make this field RAO/WI if priority-based routing is always used
- An implementation might choose to make this field RAZ/WI if priority-based routing is never used

If EL3 is present, [ICC\\_CTLR\\_EL1](#).PMHE is an alias of ICC\_CTLR\_EL3.PMHE.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**RM, bit [5]**

Routing Modifier. For legacy operation of EL1 software with [GICC\\_CTLR.FIQen](#) set to 1, this bit indicates whether interrupts can be acknowledged or observed as the Highest Priority Pending Interrupt, or whether a special INTID value is returned.

Possible values of this bit are:

RM	Meaning
0	Secure Group 0 and Non-secure Group 1 interrupts can be acknowledged and observed as the highest priority interrupt at the Secure Exception level where the interrupt is taken.
1	When accessed at EL3 in AArch64 state: <ul style="list-style-type: none"> <li>Secure Group 0 interrupts return a special INTID value of 1020. This affects accesses to <a href="#">ICC_IAR0_EL1</a> and <a href="#">ICC_HPIR0_EL1</a>.</li> <li>Non-secure Group 1 interrupts return a special INTID value of 1021. This affects accesses to <a href="#">ICC_IAR1_EL1</a> and <a href="#">ICC_HPIR1_EL1</a>.</li> </ul>

**Note**

The Routing Modifier bit is supported in AArch64 only. In systems without EL3 the behavior is as if the value is 0.

Software must ensure this bit is 0 when the Secure copy of [ICC\\_SRE\\_EL1.SRE](#) is 1, otherwise system behavior is UNPREDICTABLE.

In systems without EL3 or where the secure copy of [ICC\\_SRE\\_EL1.SRE](#) is RAO/WI, this bit is RES0.

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.

**EOImode\_EL1NS, bit [4]**

EOI mode for interrupts handled at Non-secure EL1 and EL2. Controls whether a write to an End of Interrupt register also deactivates the interrupt:

EOImode_EL1NS	Meaning
0	<a href="#">ICC_EOIR0_EL1</a> and <a href="#">ICC_EOIR1_EL1</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">ICC_DIR_EL1</a> are UNPREDICTABLE.
1	<a href="#">ICC_EOIR0_EL1</a> and <a href="#">ICC_EOIR1_EL1</a> provide priority drop functionality only. <a href="#">ICC_DIR_EL1</a> provides interrupt deactivation functionality.

If EL3 is present, [ICC\\_CTLR\\_EL1\(NS\).EOImode](#) is an alias of [ICC\\_CTLR\\_EL3.EOImode\\_EL1NS](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**EOImode\_EL1S, bit [3]**

EOI mode for interrupts handled at Secure EL1. Controls whether a write to an End of Interrupt register also deactivates the interrupt:

EOImode_EL1S	Meaning
0	<a href="#">ICC_EOIR0_EL1</a> and <a href="#">ICC_EOIR1_EL1</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">ICC_DIR_EL1</a> are UNPREDICTABLE.
1	<a href="#">ICC_EOIR0_EL1</a> and <a href="#">ICC_EOIR1_EL1</a> provide priority drop functionality only. <a href="#">ICC_DIR_EL1</a> provides interrupt deactivation functionality.

If EL3 is present, [ICC\\_CTLR\\_EL1\(S\).EOImode](#) is an alias of [ICC\\_CTLR\\_EL3.EOImode\\_EL1S](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**EOImode\_EL3, bit [2]**

EOI mode for interrupts handled at EL3. Controls whether a write to an End of Interrupt register also deactivates the interrupt:

EOImode_EL3	Meaning
0	<a href="#">ICC_EOIR0_EL1</a> and <a href="#">ICC_EOIR1_EL1</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">ICC_DIR_EL1</a> are UNPREDICTABLE.
1	<a href="#">ICC_EOIR0_EL1</a> and <a href="#">ICC_EOIR1_EL1</a> provide priority drop functionality only. <a href="#">ICC_DIR_EL1</a> provides interrupt deactivation functionality.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**CBPR\_EL1NS, bit [1]**

Common Binary Point Register, EL1 Non-secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Non-secure interrupts at EL1 and EL2:

CBPR_EL1NS	Meaning
0	<a href="#">ICC_BPR0_EL1</a> determines the preemption group for Group 0 interrupts only. <a href="#">ICC_BPR1_EL1</a> determines the preemption group for Non-secure Group 1 interrupts.
1	<a href="#">ICC_BPR0_EL1</a> determines the preemption group for Group 0 interrupts and Non-secure Group 1 interrupts. Non-secure accesses to <a href="#">GICC_BPR</a> and <a href="#">ICC_BPR1_EL1</a> access the state of <a href="#">ICC_BPR0_EL1</a> .

If EL3 is present, [ICC\\_CTLR\\_EL1\(NS\)](#).CBPR is an alias of [ICC\\_CTLR\\_EL3](#).CBPR\_EL1NS.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**CBPR\_EL1S, bit [0]**

Common Binary Point Register, EL1 Secure. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 Secure interrupts at EL1:

CBPR_EL1S	Meaning
0	<a href="#">ICC_BPR0_EL1</a> determines the preemption group for Group 0 interrupts only. <a href="#">ICC_BPR1_EL1</a> determines the preemption group for Secure Group 1 interrupts.
1	<a href="#">ICC_BPR0_EL1</a> determines the preemption group for Group 0 interrupts and Secure Group 1 interrupts. Secure EL1 accesses to <a href="#">ICC_BPR1_EL1</a> access the state of <a href="#">ICC_BPR0_EL1</a> .

If EL3 is present, [ICC\\_CTLR\\_EL1\(S\)](#).CBPR is an alias of [ICC\\_CTLR\\_EL3](#).CBPR\_EL1S.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Accessing the ICC\_CTLR\_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>	op1	CRn	CRm	op2
ICC_CTLR_EL3	110	1100	1100	100

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	-	n/a	RW
0	1	-	-	-	RW
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.

In both Security states, and not dependent on other configuration bits:

- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# ICC\_DIR\_EL1, Interrupt Controller Deactivate Interrupt Register

The ICC\_DIR\_EL1 characteristics are:

## Purpose

When interrupt priority drop is separated from interrupt deactivation, a write to this register deactivates the specified interrupt.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_DIR\_EL1 performs the same function as AArch32 System register [ICC\\_DIR](#).

## Attributes

ICC\_DIR\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_DIR\_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													INTID											

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the interrupt to be deactivated.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICC\\_CTLR\\_EL1](#).IDbits and [ICC\\_CTLR\\_EL3](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

## Accessing the ICC\_DIR\_EL1

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>	op1	CRn	CRm	op2
ICC_DIR_EL1	000	1100	1011	001

This encoding results in an access to [ICV\\_DIR\\_EL1](#) at Non-secure EL1 in the following cases:

- When HCR\_EL2.FMO is set to 1.
- When HCR\_EL2.IMO is set to 1.

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	WO	n/a	WO
x	x	1	1	-	n/a	WO	WO
x	1	0	1	-	<a href="#">ICV_DIR_EL1</a>	WO	WO
1	x	0	1	-	<a href="#">ICV_DIR_EL1</a>	WO	WO
0	0	0	1	-	WO	WO	WO

This table applies to all instructions that can access this register.

The ICC\_DIR\_EL1 register is only accessible at Non-secure EL1 when HCR\_EL2.{FMO, IMO} == {0, 0}.

### Note

At Non-secure EL1, the instruction encoding used to access ICC\_DIR\_EL1 results in an access to [ICV\\_DIR\\_EL1](#) in the following cases:

- When HCR\_EL2.FMO is set to 1.
- When HCR\_EL2.IMO is set to 1.

There are two cases when writing to [ICC\\_DIR\\_EL1](#) that were UNPREDICTABLE for a corresponding GICv2 write to [GICC\\_DIR](#):

- When EOImode == '0'. GICv3 implementations must ignore such writes. In systems supporting system error generation, an implementation might generate an SEI.
- When EOImode == '1' but no EOI has been issued. The interrupt will be de-activated by the Distributor, however the active priority in the CPU interface for the interrupt will remain set (because no EOI was issued).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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 [ICC\\_SRE\\_EL1](#).SRE==0, write accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, write accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, write accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [ICH\\_HCR\\_EL2](#).TDIR==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).FIQ==1, and [SCR\\_EL3](#).IRQ==1, Secure write accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).FIQ==1, and [SCR\\_EL3](#).IRQ==1, write 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, [SCR\\_EL3](#).IRQ==1, [HCR\\_EL2](#).IMO==0, and [HCR\\_EL2](#).FMO==0, Non-secure write accesses to this register from EL1 are trapped to EL3.



# ICC\_EOIR0\_EL1, Interrupt Controller End Of Interrupt Register 0

The ICC\_EOIR0\_EL1 characteristics are:

## Purpose

A PE writes to this register to inform the CPU interface that it has completed the processing of the specified Group 0 interrupt.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_EOIR0\_EL1 performs the same function as AArch32 System register [ICC\\_EOIR0](#).

## Attributes

ICC\_EOIR0\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_EOIR0\_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																								

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID from the corresponding [ICC\\_IAR0\\_EL1](#) access.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICC\\_CTLR\\_EL1](#).IDbits and [ICC\\_CTLR\\_EL3](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

If the EOImode bit for the current Exception level and Security state is 0, a write to this register drops the priority for the interrupt, and also deactivates the interrupt.

If the EOImode bit for the current Exception level and Security state is 1, a write to this register only drops the priority for the interrupt. Software must write to [ICC\\_DIR\\_EL1](#) to deactivate the interrupt.

The EOImode bit for the current Exception level and Security state is determined as follows:

- If EL3 is not implemented, the appropriate bit is [ICC\\_CTLR\\_EL1](#).EOImode.
- If EL3 is implemented and the software is executing at EL3, the appropriate bit is [ICC\\_CTLR\\_EL3](#).EOImode\_EL3.
- If EL3 is implemented and the software is not executing at EL3, the bit depends on the current Security state:
  - If the software is executing in Secure state, the bit is [ICC\\_CTLR\\_EL3](#).EOImode\_EL1S.
  - If the software is executing in Non-secure state, the bit is [ICC\\_CTLR\\_EL3](#).EOImode\_EL1NS.

## Accessing the ICC\_EOIR0\_EL1

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>	op1	CRn	CRm	op2
ICC_EOIR0_EL1	000	1100	1000	001

When HCR\_EL2.FMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_EOIR0\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	WO	n/a	WO
x	x	1	1	-	n/a	WO	WO
0	x	0	1	-	WO	WO	WO
1	x	0	1	-	<a href="#">ICV_EOIR0_EL1</a>	WO	WO

This table applies to all instructions that can access this register.

ICC\_EOIR0\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.FMO is set to 0.

### Note

When HCR\_EL2.FMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC\_EOIR0\_EL1 results in an access to [ICV\\_EOIR0\\_EL1](#).

A write to this register must correspond to the most recent valid read by this PE from an Interrupt Acknowledge Register, and must correspond to the INTID that was read from [ICC\\_IAR0\\_EL1](#), otherwise the system behavior is UNPREDICTABLE. A valid read is a read that returns a valid INTID that is not a special INTID.

A write of a Special INTID is ignored. See Special INTIDs, for more information.

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, write accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, write accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, write accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).FIQ==1, Secure write accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).FIQ==1, write 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\\_EL2](#).FMO==0, Non-secure write accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICC\_EOIR1\_EL1, Interrupt Controller End Of Interrupt Register 1

The ICC\_EOIR1\_EL1 characteristics are:

## Purpose

A PE writes to this register to inform the CPU interface that it has completed the processing of the specified Group 1 interrupt.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_EOIR1\_EL1 performs the same function as AArch32 System register [ICC\\_EOIR1](#).

## Attributes

ICC\_EOIR1\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_EOIR1\_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													INTID											

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID from the corresponding [ICC\\_IAR1\\_EL1](#) access.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICC\\_CTLR\\_EL1](#).IDbits and [ICC\\_CTLR\\_EL3](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

If the EOImode bit for the current Exception level and Security state is 0, a write to this register drops the priority for the interrupt, and also deactivates the interrupt.

If the EOImode bit for the current Exception level and Security state is 1, a write to this register only drops the priority for the interrupt. Software must write to [ICC\\_DIR\\_EL1](#) to deactivate the interrupt.

The EOImode bit for the current Exception level and Security state is determined as follows:

- If EL3 is not implemented, the appropriate bit is [ICC\\_CTLR\\_EL1](#).EOImode.
- If EL3 is implemented and the software is executing at EL3, the appropriate bit is [ICC\\_CTLR\\_EL3](#).EOImode\_EL3.
- If EL3 is implemented and the software is not executing at EL3, the bit depends on the current Security state:
  - If the software is executing in Secure state, the bit is [ICC\\_CTLR\\_EL3](#).EOImode\_EL1S.
  - If the software is executing in Non-secure state, the bit is [ICC\\_CTLR\\_EL3](#).EOImode\_EL1NS.

## Accessing the ICC\_EOIR1\_EL1

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>	op1	CRn	CRm	op2
ICC_EOIR1_EL1	000	1100	1100	001

When HCR\_EL2.IMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_EOIR1\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	WO	n/a	WO
x	x	1	1	-	n/a	WO	WO
x	0	0	1	-	WO	WO	WO
x	1	0	1	-	<a href="#">ICV_EOIR1_EL1</a>	WO	WO

This table applies to all instructions that can access this register.

ICC\_EOIR1\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.IMO is set to 0.

### Note

When HCR\_EL2.IMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC\_EOIR1\_EL1 results in an access to [ICV\\_EOIR1\\_EL1](#).

A write to this register must correspond to the most recent valid read by this PE from an Interrupt Acknowledge Register, and must correspond to the INTID that was read from [ICC\\_IARI\\_EL1](#), otherwise the system behavior is UNPREDICTABLE. A valid read is a read that returns a valid INTID that is not a special INTID.

A write of a Special INTID is ignored. See Special INTIDs, for more information.

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, write accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, write accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, write accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).IRQ==1, Secure write accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).IRQ==1, write 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\\_EL2](#).IMO==0, Non-secure write accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICC\_HPPIR0\_EL1, Interrupt Controller Highest Priority Pending Interrupt Register 0

The ICC\_HPPIR0\_EL1 characteristics are:

## Purpose

Indicates the highest priority pending Group 0 interrupt on the CPU interface.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_HPPIR0\_EL1 performs the same function as AArch32 System register [ICC\\_HPPIR0](#).

## Attributes

ICC\_HPPIR0\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_HPPIR0\_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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the highest priority pending interrupt, if that interrupt is observable at the current Security state and Exception level.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. These special INTIDs can be one of: 1020, 1021, or 1023. See Special INTIDs, for more information.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICC\\_CTLR\\_EL1](#).IDbits and [ICC\\_CTLR\\_EL3](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

## Accessing the ICC\_HPPIR0\_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>	op1	CRn	CRm	op2
ICC_HPPIR0_EL1	000	1100	1000	010

When HCR\_EL2.FMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_HPPIR0\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	RO	n/a	RO
x	x	1	1	-	n/a	RO	RO
0	x	0	1	-	RO	RO	RO
1	x	0	1	-	<a href="#">ICV_HPPIR0_EL1</a>	RO	RO

This table applies to all instructions that can access this register.

ICC\_HPPIR0\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.FMO is set to 0.

### Note

When HCR\_EL2.FMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC\_HPPIR0\_EL1 results in an access to [ICV\\_HPPIR0\\_EL1](#).

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, read accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, read accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, read accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).FIQ==1, Secure read accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).FIQ==1, read 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\\_EL2](#).FMO==0, Non-secure read accesses to this register from EL1 are trapped to EL3.

# ICC\_HPPIR1\_EL1, Interrupt Controller Highest Priority Pending Interrupt Register 1

The ICC\_HPPIR1\_EL1 characteristics are:

## Purpose

Indicates the highest priority pending Group 1 interrupt on the CPU interface.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_HPPIR1\_EL1 performs the same function as AArch32 System register [ICC\\_HPPIR1](#).

## Attributes

ICC\_HPPIR1\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_HPPIR1\_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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the highest priority pending interrupt, if that interrupt is observable at the current Security state and Exception level.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. These special INTIDs can be one of: 1020, 1021, or 1023. See [Special INTIDs](#), for more information.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICC\\_CTLR\\_EL1](#).IDbits and [ICC\\_CTLR\\_EL3](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

## Accessing the ICC\_HPPIR1\_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>	op1	CRn	CRm	op2
ICC_HPPIR1_EL1	000	1100	1100	010

When HCR\_EL2.IMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_HPPIR1\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	RO	n/a	RO
x	x	1	1	-	n/a	RO	RO
x	0	0	1	-	RO	RO	RO
x	1	0	1	-	<a href="#">ICV_HPPIR1_EL1</a>	RO	RO

This table applies to all instructions that can access this register.

ICC\_HPPIR1\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.IMO is set to 0.

### Note

When HCR\_EL2.IMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC\_HPPIR1\_EL1 results in an access to [ICV\\_HPPIR1\\_EL1](#).

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, read accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, read accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, read accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).IRQ==1, Secure read accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).IRQ==1, read 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\\_EL2](#).IMO==0, Non-secure read accesses to this register from EL1 are trapped to EL3.

# ICC\_IAR0\_EL1, Interrupt Controller Interrupt Acknowledge Register 0

The ICC\_IAR0\_EL1 characteristics are:

## Purpose

The PE reads this register to obtain the INTID of the signaled Group 0 interrupt. This read acts as an acknowledge for the interrupt.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_IAR0\_EL1 performs the same function as AArch32 System register [ICC\\_IAR0](#).

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when  $PSTATE.\{I,F\} == \{0,0\}$ ). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. See Observability of the effects of accesses to the GIC registers, for more information.

## Attributes

ICC\_IAR0\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_IAR0\_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													INTID											

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the signaled interrupt.

This is the INTID of the highest priority pending interrupt, if that interrupt is of sufficient priority for it to be signaled to the PE, and if it can be acknowledged at the current Security state and Exception level.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. These special INTIDs can be one of: 1020, 1021, or 1023. See Special INTIDs, for more information.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICC\\_CTLR\\_EL1](#).IDbits and [ICC\\_CTLR\\_EL3](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

## Accessing the ICC\_IAR0\_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>	op1	CRn	CRm	op2
ICC_IAR0_EL1	000	1100	1000	000

When HCR\_EL2.FMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_IAR0\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	RO	n/a	RO
x	x	1	1	-	n/a	RO	RO
0	x	0	1	-	RO	RO	RO
1	x	0	1	-	<a href="#">ICV_IAR0_EL1</a>	RO	RO

This table applies to all instructions that can access this register.

ICC\_IAR0\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.FMO is set to 0.

### Note

When HCR\_EL2.FMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC\_IAR0\_EL1 results in an access to [ICV\\_IAR0\\_EL1](#).

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, read accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, read accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, read accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).FIQ==1, Secure read accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).FIQ==1, read 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\\_EL2](#).FMO==0, Non-secure read accesses to this register from EL1 are trapped to EL3.





# ICC\_IAR1\_EL1, Interrupt Controller Interrupt Acknowledge Register 1

The ICC\_IAR1\_EL1 characteristics are:

## Purpose

The PE reads this register to obtain the INTID of the signaled Group 1 interrupt. This read acts as an acknowledge for the interrupt.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_IAR1\_EL1 performs the same function as AArch32 System register [ICC\\_IAR1](#).

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when  $PSTATE.\{I,F\} == \{0,0\}$ ). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. See Observability of the effects of accesses to the GIC registers, for more information.

## Attributes

ICC\_IAR1\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_IAR1\_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													INTID											

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the signaled interrupt.

This is the INTID of the highest priority pending interrupt, if that interrupt is of sufficient priority for it to be signaled to the PE, and if it can be acknowledged at the current Security state and Exception level.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. These special INTIDs can be one of: 1020, 1021, or 1023. See Special INTIDs, for more information.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICC\\_CTLR\\_EL1](#).IDbits and [ICC\\_CTLR\\_EL3](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

## Accessing the ICC\_IAR1\_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>	op1	CRn	CRm	op2
ICC_IAR1_EL1	000	1100	1100	000

When HCR\_EL2.IMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_IAR1\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	RO	n/a	RO
x	x	1	1	-	n/a	RO	RO
x	0	0	1	-	RO	RO	RO
x	1	0	1	-	<a href="#">ICV_IAR1_EL1</a>	RO	RO

This table applies to all instructions that can access this register.

ICC\_IAR1\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.IMO is set to 0.

### Note

When HCR\_EL2.IMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC\_IAR1\_EL1 results in an access to [ICV\\_IAR1\\_EL1](#).

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, read accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, read accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, read accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).IRQ==1, Secure read accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).IRQ==1, read 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\\_EL2](#).IMO==0, Non-secure read accesses to this register from EL1 are trapped to EL3.



# ICC\_IGRPEN0\_EL1, Interrupt Controller Interrupt Group 0 Enable register

The ICC\_IGRPEN0\_EL1 characteristics are:

## Purpose

Controls whether Group 0 interrupts are enabled or not.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_IGRPEN0\_EL1 is architecturally mapped to AArch32 System register [ICC\\_IGRPEN0](#).

## Attributes

ICC\_IGRPEN0\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_IGRPEN0\_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	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\\_EL2.VENG0](#).

If the highest priority pending interrupt for that PE is a Group 0 interrupt using 1 of N model, then the interrupt will be targeted to another PE as a result of the Enable bit changing from 1 to 0.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the ICC\_IGRPEN0\_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>	op1	CRn	CRm	op2
ICC_IGRPEN0_EL1	000	1100	1100	110

When HCR\_EL2.FMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_IGRPEN0\\_EL1](#).

## 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	-	<a href="#">ICV_IGRPEN0_EL1</a>	RW	RW

This table applies to all instructions that can access this register.

ICC\_IGRPEN0\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.FMO is set to 0.

### Note

When HCR\_EL2.FMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC\_IGRPEN0\_EL1 results in an access to [ICV\\_IGRPEN0\\_EL1](#).

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- 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 AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).FIQ==1, Secure accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).FIQ==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.FIQ](#)=1, and [HCR\\_EL2.FMO](#)=0, Non-secure accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICC\_IGRPEN1\_EL1, Interrupt Controller Interrupt Group 1 Enable register

The ICC\_IGRPEN1\_EL1 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

AArch64 System register ICC\_IGRPEN1\_EL1 (S) is architecturally mapped to AArch32 System register [ICC\\_IGRPEN1 \(S\)](#).

AArch64 System register ICC\_IGRPEN1\_EL1 (NS) is architecturally mapped to AArch32 System register [ICC\\_IGRPEN1 \(NS\)](#).

## Attributes

ICC\_IGRPEN1\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_IGRPEN1\_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	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\\_EL2](#).VENG1.

If EL3 is present:

- The Secure [ICC\\_IGRPEN1\\_EL1](#).Enable bit is a read/write alias of the [ICC\\_IGRPEN1\\_EL3](#).EnableGrp1S bit.
- The Non-secure [ICC\\_IGRPEN1\\_EL1](#).Enable bit is a read/write alias of the [ICC\\_IGRPEN1\\_EL3](#).EnableGrp1NS bit.

If the highest priority pending interrupt for that PE is a Group 1 interrupt using 1 of N model, then the interrupt will target another PE as a result of the Enable bit changing from 1 to 0.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the ICC\_IGRPEN1\_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>	op1	CRn	CRm	op2
ICC_IGRPEN1_EL1	000	1100	1100	111

When HCR\_EL2.IMO is set to 1, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_IGRPEN1\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility				Instance
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3	
x	x	x	0	-	RW	n/a	RW	ICC_IGRPEN1_EL1_s
x	x	1	1	-	n/a	RW	RW	ICC_IGRPEN1_EL1_ns
x	0	0	1	-	RW	RW	RW	ICC_IGRPEN1_EL1_ns
x	1	0	1	-	<a href="#">ICV_IGRPEN1_EL1</a>	RW	RW	ICC_IGRPEN1_EL1_ns

This table applies to all instructions that can access this register.

ICC\_IGRPEN1\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.IMO is set to 0.

### Note

When HCR\_EL2.IMO is set to 1, at Non-secure EL1, the instruction encoding used to access ICC\_IGRPEN1\_EL1 results in an access to [ICV\\_IGRPEN1\\_EL1](#).

If EL3 is present and this register is accessed at EL3, the copy of this register appropriate to the current setting of SCR\_EL3.NS is accessed.

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- 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 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\\_EL2](#).IMO==0, Non-secure accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICC\_IGRPEN1\_EL3, Interrupt Controller Interrupt Group 1 Enable register (EL3)

The ICC\_IGRPEN1\_EL3 characteristics are:

## Purpose

Controls whether Group 1 interrupts are enabled or not.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_IGRPEN1\_EL3 can be mapped to AArch32 System register [ICC\\_MGRPEN1](#), but this is not architecturally mandated.

## Attributes

ICC\_IGRPEN1\_EL3 is a 32-bit register.

## Field descriptions

The ICC\_IGRPEN1\_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
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	EnableGrp1S	EnableGrp1NS

### Bits [31:2]

Reserved, RES0.

### EnableGrp1S, bit [1]

Enables Group 1 interrupts for the Secure state.

EnableGrp1S	Meaning
0	Secure Group 1 interrupts are disabled.
1	Secure Group 1 interrupts are enabled.

The Secure [ICC\\_IGRPEN1\\_EL1](#).Enable bit is a read/write alias of the ICC\_IGRPEN1\_EL3.EnableGrp1S bit.

If the highest priority pending interrupt for that PE is a Group 1 interrupt using 1 of N model, then the interrupt will target another PE as a result of the Enable bit changing from 1 to 0.

When this register has an architecturally-defined reset value, this field resets to 0.

### EnableGrp1NS, bit [0]

Enables Group 1 interrupts for the Non-secure state.

EnableGrp1NS	Meaning
0	Non-secure Group 1 interrupts are disabled.
1	Non-secure Group 1 interrupts are enabled.

The Non-secure [ICC\\_IGRPEN1\\_EL1](#).Enable bit is a read/write alias of the ICC\_IGRPEN1\_EL3.EnableGrp1NS bit.

If the highest priority pending interrupt for that PE is a Group 1 interrupt using 1 of N model, then the interrupt will target another PE as a result of the Enable bit changing from 1 to 0.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the ICC\_IGRPEN1\_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>	op1	CRn	CRm	op2
ICC_IGRPEN1_EL3	110	1100	1100	111

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	-	n/a	RW
x	x	0	1	-	-	-	RW
x	x	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.

In both Security states, and not dependent on other configuration bits:

- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.

# ICC\_PMR\_EL1, Interrupt Controller Interrupt Priority Mask Register

The ICC\_PMR\_EL1 characteristics are:

## Purpose

Provides an interrupt priority filter. Only interrupts with a higher priority than the value in this register are signaled to the PE.

Writes to this register must be high performance and must ensure that no interrupt of lower priority than the written value occurs after the write, without requiring an ISB or an exception boundary.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_PMR\_EL1 is architecturally mapped to AArch32 System register [ICC\\_PMR](#).

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that writes to this register are self-synchronising. This ensures that no interrupts below the written PMR value will be taken after a write to this register is architecturally executed. See Observability of the effects of accesses to the GIC registers, for more information.

## Attributes

ICC\_PMR\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_PMR\_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	Priority							

### Bits [31:8]

Reserved, RES0.

### Priority, bits [7:0]

The priority mask level for the CPU interface. If the priority of an interrupt is higher than the value indicated by this field, the interface signals the interrupt to the PE.

The possible priority field values are as follows:

Implemented priority bits	Possible priority field values	Number of priority levels
[7:0]	0x00-0xFF (0-255), all values	256
[7:1]	0x00-0xFE (0-254), even values only	128
[7:2]	0x00-0xFC (0-252), in steps of 4	64
[7:3]	0x00-0xF8 (0-248), in steps of 8	32
[7:4]	0x00-0xF0 (0-240), in steps of 16	16

Unimplemented priority bits are RAZ/WI.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the ICC\_PMR\_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>	op1	CRn	CRm	op2
ICC_PMR_EL1	000	0100	0110	000

When HCR\_EL2.{FMO, IMO} != {0, 0}, execution of this encoding at Non-secure EL1 results in an access to [ICV\\_PMR\\_EL1](#).

## 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
x	1	0	1	-	<a href="#">ICV_PMR_EL1</a>	RW	RW
1	x	0	1	-	<a href="#">ICV_PMR_EL1</a>	RW	RW
0	0	0	1	-	RW	RW	RW

This table applies to all instructions that can access this register.

ICC\_PMR\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.{FMO, IMO} == {0, 0}.

### Note

When HCR\_EL2.{FMO, IMO} != {0, 0}, at Non-secure EL1, the instruction encoding used to access ICC\_PMR\_EL1 results in an access to [ICV\\_PMR\\_EL1](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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 [ICC\\_SRE\\_EL1](#).SRE==0, accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).FIQ==1, and [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.FIQ](#)=1, and [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.FIQ](#)=1, [SCR\\_EL3.IRQ](#)=1, [HCR\\_EL2.IMO](#)=0, and [HCR\\_EL2.FMO](#)=0, Non-secure accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICC\_RPR\_EL1, Interrupt Controller Running Priority Register

The ICC\_RPR\_EL1 characteristics are:

## Purpose

Indicates the Running priority of the CPU interface.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_RPR\_EL1 performs the same function as AArch32 System register [ICC\\_RPR](#).

## Attributes

ICC\_RPR\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_RPR\_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	Priority									

### Bits [31:8]

Reserved, RES0.

### Priority, bits [7:0]

The current running priority on the CPU interface. This is the group priority of the current active interrupt.

If there are no active interrupts on the CPU interface, or all active interrupts have undergone a priority drop, the value returned is the Idle priority.

The priority returned is the group priority as if the BPR for the current Exception level and Security state was set to the minimum value of BPR for the number of implemented priority bits.

---

#### Note

If 8 bits of priority are implemented the group priority is bits[7:1] of the priority.

---

## Accessing the ICC\_RPR\_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>	op1	CRn	CRm	op2
ICC_RPR_EL1	000	1100	1011	011

When  $\text{HCR\_EL2}\{FMO, IMO\} \neq \{0, 0\}$ , execution of this encoding at Non-secure EL1 results in an access to [ICV\\_RPR\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	RO	n/a	RO
x	x	1	1	-	n/a	RO	RO
x	1	0	1	-	<a href="#">ICV_RPR_EL1</a>	RO	RO
1	x	0	1	-	<a href="#">ICV_RPR_EL1</a>	RO	RO
0	0	0	1	-	RO	RO	RO

This table applies to all instructions that can access this register.

ICC\_RPR\_EL1 is only accessible at Non-secure EL1 when  $\text{HCR\_EL2}\{FMO, IMO\} = \{0, 0\}$ .

### Note

When  $\text{HCR\_EL2}\{FMO, IMO\} \neq \{0, 0\}$ , at Non-secure EL1, the instruction encoding used to access ICC\_RPR\_EL1 results in an access to [ICV\\_RPR\\_EL1](#).

Software cannot determine the number of implemented priority bits from a read of 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 [ICC\\_SRE\\_EL1](#).SRE==0, read accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, read accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, read accesses to this register from EL3 are trapped to EL3.

When  $\text{SCR\_EL3.NS}=1$  :

- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=0$  :

- If [SCR\\_EL3](#).FIQ==1, and [SCR\\_EL3](#).IRQ==1, Secure read accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).FIQ==1, and [SCR\\_EL3](#).IRQ==1, read accesses to this register from EL2 are trapped to EL3.

When EL3 is implemented and is using AArch64 and  $\text{SCR\_EL3.NS}=1$  :

- If [SCR\\_EL3](#).FIQ==1, [SCR\\_EL3](#).IRQ==1, [HCR\\_EL2](#).IMO==0, and [HCR\\_EL2](#).FMO==0, Non-secure read accesses to this register from EL1 are trapped to EL3.



# ICC\_SGI0R\_EL1, Interrupt Controller Software Generated Interrupt Group 0 Register

The ICC\_SGI0R\_EL1 characteristics are:

## Purpose

Generates Secure Group 0 SGIs.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_SGI0R\_EL1 performs the same function as AArch32 System register [ICC\\_SGI0R](#).

## Attributes

ICC\_SGI0R\_EL1 is a 64-bit register.

## Field descriptions

The ICC\_SGI0R\_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	Aff3								0	0	0	0	0	0	0	IRM	Aff2									
0	0	0	0	INTID				Aff1								TargetList																	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

### Bits [63:56]

Reserved, RES0.

### Aff3, bits [55:48]

The affinity 3 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

### Bits [47:41]

Reserved, RES0.

### IRM, bit [40]

Interrupt Routing Mode. Determines how the generated interrupts are distributed to PEs. Possible values are:

IRM	Meaning
0	Interrupts routed to the PEs specified by Aff3.Aff2.Aff1.<target list>.
1	Interrupts routed to all PEs in the system, excluding "self".

**Aff2, bits [39:32]**

The affinity 2 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

**Bits [31:28]**

Reserved, RES0.

**INTID, bits [27:24]**

The INTID of the SGI.

**Aff1, bits [23:16]**

The affinity 1 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

**TargetList, bits [15:0]**

Target List. The set of PEs for which SGI interrupts will be generated. Each bit corresponds to the PE within a cluster with an Affinity 0 value equal to the bit number.

If a bit is 1 and the bit does not correspond to a valid target PE, the bit must be ignored by the Distributor. It is IMPLEMENTATION DEFINED whether, in such cases, a Distributor can signal a system error.

**Note**

This restricts a system to sending targeted SGIs to PEs with an affinity 0 number that is less than 16.

If SRE is set only for Secure EL3, software executing at EL3 might use the System register interface to generate SGIs. Therefore, the Distributor must always be able to receive and acknowledge Generate SGI packets received from CPU interface regardless of the ARE settings for a Security state. However, the Distributor might discard such packets.

If the IRM bit is 1, this field is RES0.

**Accessing the ICC\_SGI0R\_EL1**

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>	op1	CRn	CRm	op2
ICC_SGI0R_EL1	000	1100	1011	111

**Accessibility**

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	WO	n/a	WO
0	1	-	WO	WO	WO
1	1	-	n/a	WO	WO

This table applies to all instructions that can access this register.

This register allows software executing in a Secure state to generate Group 0 SGIs. It will also allow software executing in a Non-secure state to generate Group 0 SGIs, if permitted by the settings of [GICR\\_NSACR](#) in the Redistributor corresponding to the target PE.

When [GICD\\_CTLR.DS](#)==0, Non-secure writes do not generate an interrupt for a target PE if not permitted by the [GICR\\_NSACR](#) register associated with the target PE. For more information see Use of control registers for SGI forwarding.

---

#### Note

Accesses at EL3 are treated as Secure regardless of the value of SCR\_EL3.NS.

---

## 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 [ICC\\_SRE\\_EL1.SRE](#)==0, write accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2.SRE](#)==0, write accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3.SRE](#)==0, write accesses to this register from EL3 are trapped to EL3.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HCR\\_EL2.FMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR\\_EL2.IMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HCR\\_EL2.FMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR\\_EL2.IMO](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2.TC](#)==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3.FIQ](#)==1, and [SCR\\_EL3.IRQ](#)==1, Secure write accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3.FIQ](#)==1, and [SCR\\_EL3.IRQ](#)==1, write 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, [SCR\\_EL3.IRQ](#)==1, [HCR\\_EL2.IMO](#)==0, and [HCR\\_EL2.FMO](#)==0, Non-secure write accesses to this register from EL1 are trapped to EL3.

# ICC\_SGI1R\_EL1, Interrupt Controller Software Generated Interrupt Group 1 Register

The ICC\_SGI1R\_EL1 characteristics are:

## Purpose

Generates Group 1 SGIs for the current Security state.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_SGI1R\_EL1 performs the same function as AArch32 System register [ICC\\_SGI1R](#).

Under certain conditions a write to ICC\_SGI1R\_EL1 can generate Group 0 interrupts, see Forwarding an SGI to a target PE.

## Attributes

ICC\_SGI1R\_EL1 is a 64-bit register.

## Field descriptions

The ICC\_SGI1R\_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	Aff3								0	0	0	0	0	0	0	0	IRM	Aff2									
0	0	0	0	INTID				Aff1								TargetList																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			

### Bits [63:56]

Reserved, RES0.

### Aff3, bits [55:48]

The affinity 3 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

### Bits [47:41]

Reserved, RES0.

### IRM, bit [40]

Interrupt Routing Mode. Determines how the generated interrupts are distributed to PEs. Possible values are:

IRM	Meaning
0	Interrupts routed to the PEs specified by Aff3.Aff2.Aff1.<target list>.
1	Interrupts routed to all PEs in the system, excluding "self".

**Aff2, bits [39:32]**

The affinity 2 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

**Bits [31:28]**

Reserved, RES0.

**INTID, bits [27:24]**

The INTID of the SGI.

**Aff1, bits [23:16]**

The affinity 1 value of the affinity path of the cluster for which SGI interrupts will be generated.

If the IRM bit is 1, this field is RES0.

**TargetList, bits [15:0]**

Target List. The set of PEs for which SGI interrupts will be generated. Each bit corresponds to the PE within a cluster with an Affinity 0 value equal to the bit number.

If a bit is 1 and the bit does not correspond to a valid target PE, the bit must be ignored by the Distributor. It is IMPLEMENTATION DEFINED whether, in such cases, a Distributor can signal a system error.

**Note**

This restricts a system to sending targeted SGIs to PEs with an affinity 0 number that is less than 16.

If SRE is set only for Secure EL3, software executing at EL3 might use the System register interface to generate SGIs. Therefore, the Distributor must always be able to receive and acknowledge Generate SGI packets received from CPU interface regardless of the ARE settings for a Security state. However, the Distributor might discard such packets.

If the IRM bit is 1, this field is RES0.

## Accessing the ICC\_SGI1R\_EL1

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>	op1	CRn	CRm	op2
ICC_SGI1R_EL1	000	1100	1011	101

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	WO	n/a	WO
0	1	-	WO	WO	WO
1	1	-	n/a	WO	WO

This table applies to all instructions that can access this register.

## Note

Accesses at EL3 are treated as Secure regardless of the value of SCR\_EL3.NS.

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, write accesses to this register from EL1 are trapped to EL1.
- If [ICC\\_SRE\\_EL2](#).SRE==0, write accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, write accesses to this register from EL3 are trapped to EL3.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==0 :

- If [HCR\\_EL2](#).FMO==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR\\_EL2](#).IMO==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && .E2H==1 && HCR\_EL2.TGE==0 :

- If [HCR\\_EL2](#).FMO==1, Non-secure write accesses to this register from EL1 are trapped to EL2.
- If [HCR\\_EL2](#).IMO==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==0 :

- If [SCR\\_EL3](#).FIQ==1, and [SCR\\_EL3](#).IRQ==1, Secure write accesses to this register from EL1 are trapped to EL3.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3](#).FIQ==1, and [SCR\\_EL3](#).IRQ==1, write 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, [SCR\\_EL3](#).IRQ==1, [HCR\\_EL2](#).IMO==0, and [HCR\\_EL2](#).FMO==0, Non-secure write accesses to this register from EL1 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICC\_SRE\_EL1, Interrupt Controller System Register Enable register (EL1)

The ICC\_SRE\_EL1 characteristics are:

## Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL1.

This register is part of:

- The GIC system registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_SRE\_EL1 (S) is architecturally mapped to AArch32 System register [ICC\\_SRE \(S\)](#).

AArch64 System register ICC\_SRE\_EL1 (NS) is architecturally mapped to AArch32 System register [ICC\\_SRE \(NS\)](#).

## Attributes

ICC\_SRE\_EL1 is a 32-bit register.

## Field descriptions

The ICC\_SRE\_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	DIB	DFB	SRE

### Bits [31:3]

Reserved, RES0.

### DIB, bit [2]

Disable IRQ bypass.

DIB	Meaning
0	IRQ bypass enabled.
1	IRQ bypass disabled.

If EL3 is implemented and [GICD\\_CTLR](#).DS == 0, this field is a read-only alias of [ICC\\_SRE\\_EL3](#).DIB.

If EL3 is implemented and [GICD\\_CTLR](#).DS == 1, and EL2 is not implemented, this field is a read-write alias of [ICC\\_SRE\\_EL3](#).DIB.

If EL3 is not implemented or [GICD\\_CTLR](#).DS == 1, and EL2 is implemented, this field is a read-only alias of [ICC\\_SRE\\_EL2](#).DIB.

In systems that do not support IRQ bypass, this field is RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

### DFB, bit [1]

Disable FIQ bypass.

DFB	Meaning
0	FIQ bypass enabled.
1	FIQ bypass disabled.

If EL3 is implemented and [GICD\\_CTLR.DS](#) == 0, this field is a read-only alias of [ICC\\_SRE\\_EL3.DFB](#).

If EL3 is implemented and [GICD\\_CTLR.DS](#) == 1, and EL2 is not implemented, this field is a read-write alias of [ICC\\_SRE\\_EL3.DFB](#).

If EL3 is not implemented or [GICD\\_CTLR.DS](#) == 1, and EL2 is implemented, this field is a read-only alias of [ICC\\_SRE\\_EL2.DFB](#).

In systems that do not support FIQ bypass, this field is RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

## SRE, bit [0]

System Register Enable.

SRE	Meaning
0	The memory-mapped interface must be used. Access at EL1 to any ICC_* System register other than ICC_SRE_EL1 is trapped to EL1.
1	The System register interface for the current Security state is enabled.

If software changes this bit from 1 to 0 in the Secure instance of this register, the results are UNPREDICTABLE.

If an implementation supports only a System register interface to the GIC CPU interface, this bit is RAO/WI.

If EL3 is implemented and [ICC\\_SRE\\_EL3.SRE](#)==0 the Secure copy of this bit is RAZ/WI.

If EL2 is implemented and [ICC\\_SRE\\_EL2.SRE](#)==0 the Non-secure copy of this bit is RAZ/WI.

If EL3 is implemented and [ICC\\_SRE\\_EL3.SRE](#)==0 the Non-secure copy of this bit is RAZ/WI.

GICv3 implementations that do not require GICv2 compatibility might choose to make this bit RAO/WI. The following options are supported:

- The Non-secure copy of [ICC\\_SRE\\_EL1.SRE](#) can be RAO/WI if [ICC\\_SRE\\_EL2.SRE](#) is also RAO/WI. This means all Non-secure software, including VMs using only virtual interrupts, must access the GIC using System registers.
- The Secure copy of [ICC\\_SRE\\_EL1.SRE](#) can be RAO/WI if [ICC\\_SRE\\_EL3.SRE](#) and [ICC\\_SRE\\_EL2.SRE](#) are also RAO/WI. This means that all Secure software must access the GIC using System registers and all Non-secure accesses to registers for physical interrupts must use System registers.

### Note

A VM using only virtual interrupts might still use memory-mapped access if the Non-secure copy of [ICC\\_SRE\\_EL1.SRE](#) is not RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

## Accessing the ICC\_SRE\_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>	op1	CRn	CRm	op2
ICC_SRE_EL1	000	1100	1100	101

## Accessibility

The register is accessible as follows:



Control		Accessibility				Instance
TGE	NS	EL0	EL1	EL2	EL3	
x	0	-	RW	n/a	RW	ICC_SRE_EL1_s
0	1	-	RW	RW	RW	ICC_SRE_EL1_ns
1	1	-	n/a	RW	RW	ICC_SRE_EL1_ns

This table applies to all instructions that can access this register.

Execution with [ICC\\_SRE\\_EL1](#).SRE set to 0 might make some System registers UNKNOWN.

## 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 [ICC\\_SRE\\_EL3](#).Enable==0, and EL3 is implemented, accesses to this register from EL1 and EL2 are trapped to EL3.

When SCR\_EL3.NS==1 :

- If [ICC\\_SRE\\_EL2](#).Enable==0, and EL2 is implemented, Non-secure accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICC\_SRE\_EL2, Interrupt Controller System Register Enable register (EL2)

The ICC\_SRE\_EL2 characteristics are:

## Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL2.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_SRE\_EL2 is architecturally mapped to AArch32 System register [ICC\\_HSRE](#).

If EL2 is not implemented, this register is RES0 from EL3.

## Attributes

ICC\_SRE\_EL2 is a 32-bit register.

## Field descriptions

The ICC\_SRE\_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
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	DIB	DFB	SRE

### Bits [31:4]

Reserved, RES0.

### Enable, bit [3]

Enable. Enables lower Exception level access to [ICC\\_SRE\\_EL1](#).

Enable	Meaning
0	Non-secure EL1 accesses to <a href="#">ICC_SRE_EL1</a> trap to EL2.
1	Non-secure EL1 accesses to <a href="#">ICC_SRE_EL1</a> do not trap to EL2.

If ICC\_SRE\_EL2.SRE is RAO/WI, an implementation is permitted to make the Enable bit RAO/WI.

If ICC\_SRE\_EL2.SRE is 0, the Enable bit behaves as 1 for all purposes other than reading the value of the bit.

### DIB, bit [2]

Disable IRQ bypass.

DIB	Meaning
0	IRQ bypass enabled.
1	IRQ bypass disabled.

If EL3 is implemented and [GICD\\_CTLR](#).DS is 0, this field is a read-only alias of [ICC\\_SRE\\_EL3](#).DIB.

If EL3 is implemented and [GICD\\_CTLR.DS](#) is 1, this field is a read-write alias of [ICC\\_SRE\\_EL3.DIB](#).

In systems that do not support IRQ bypass, 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 0.

### DFB, bit [1]

Disable FIQ bypass.

DFB	Meaning
0	FIQ bypass enabled.
1	FIQ bypass disabled.

If EL3 is implemented and [GICD\\_CTLR.DS](#) is 0, this field is a read-only alias of [ICC\\_SRE\\_EL3.DFB](#).

If EL3 is implemented and [GICD\\_CTLR.DS](#) is 1, this field is a read-write alias of [ICC\\_SRE\\_EL3.DFB](#).

In systems that do not support FIQ bypass, 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 0.

### SRE, bit [0]

System Register Enable.

SRE	Meaning
0	The memory-mapped interface must be used. Access at EL2 to any ICH_* or ICC_* register other than <a href="#">ICC_SRE_EL1</a> or ICC_SRE_EL2, is trapped to EL2.
1	The System register interface to the ICH_* registers and the EL1 and EL2 ICC_* registers is enabled for EL2.

If software changes this bit from 1 to 0, the results are UNPREDICTABLE.

If an implementation supports only a System register interface to the GIC CPU interface, this bit is RAO/WI.

If EL3 is implemented and ICC\_SRE\_EL3.SRE==0 this bit is RAZ/WI.

GICv3 implementations that do not require GICv2 compatibility might choose to make this bit RAO/WI, but this is only allowed if ICC\_SRE\_EL3.SRE is also RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

## Accessing the ICC\_SRE\_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>	op1	CRn	CRm	op2
ICC_SRE_EL2	100	1100	1001	101

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3

x	0	-	-	n/a	-
0	1	-	-	RW	RW
1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Execution with [ICC\\_SRE\\_EL2](#).SRE set to 0 might make some System registers UNKNOWN.

## 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 [ICC\\_SRE\\_EL3](#).Enable==0, accesses to this register from EL2 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICC\_SRE\_EL3, Interrupt Controller System Register Enable register (EL3)

The ICC\_SRE\_EL3 characteristics are:

## Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL3.

This register is part of:

- The GIC system registers functional group.
- The Security registers functional group.
- The GIC control registers functional group.

## Configuration

AArch64 System register ICC\_SRE\_EL3 can be mapped to AArch32 System register [ICC\\_MSRE](#), but this is not architecturally mandated.

## Attributes

ICC\_SRE\_EL3 is a 32-bit register.

## Field descriptions

The ICC\_SRE\_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
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	DIB	DFB	SRE

### Bits [31:4]

Reserved, RES0.

### Enable, bit [3]

Enable. Enables lower Exception level access to [ICC\\_SRE\\_EL1](#) and [ICC\\_SRE\\_EL2](#).

Enable	Meaning
0	Secure EL1 accesses to Secure <a href="#">ICC_SRE_EL1</a> trap to EL3. EL2 accesses to Non-secure <a href="#">ICC_SRE_EL1</a> and <a href="#">ICC_SRE_EL2</a> trap to EL3. Non-secure EL1 accesses to <a href="#">ICC_SRE_EL1</a> trap to EL3, unless these accesses are trapped to EL2 as a result of ICC_SRE_EL3.Enable == 0.
1	Secure EL1 accesses to Secure <a href="#">ICC_SRE_EL1</a> do not trap to EL3. EL2 accesses to Non-secure <a href="#">ICC_SRE_EL1</a> and <a href="#">ICC_SRE_EL2</a> do not trap to EL3. Non-secure EL1 accesses to <a href="#">ICC_SRE_EL1</a> do not trap to EL3.

If ICC\_SRE\_EL3.SRE is RAO/WI, an implementation is permitted to make the Enable bit RAO/WI.

If ICC\_SRE\_EL3.SRE is 0, the Enable bit behaves as 1 for all purposes other than reading the value of the bit.

### DIB, bit [2]

Disable IRQ bypass.

DIB	Meaning
0	IRQ bypass enabled.
1	IRQ bypass disabled.

In systems that do not support IRQ bypass, 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 0.

### DFB, bit [1]

Disable FIQ bypass.

DFB	Meaning
0	FIQ bypass enabled.
1	FIQ bypass disabled.

In systems that do not support FIQ bypass, 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 0.

### SRE, bit [0]

System Register Enable.

SRE	Meaning
0	The memory-mapped interface must be used. Access at EL3 to any ICH_* or ICC_* register other than <a href="#">ICC_SRE_EL1</a> , <a href="#">ICC_SRE_EL2</a> , or ICC_SRE_EL3 is trapped to EL3
1	The System register interface to the ICH_* registers and the EL1, EL2, and EL3 ICC_* registers is enabled for EL3.

If software changes this bit from 1 to 0, the results are UNPREDICTABLE.

GICv3 implementations that do not require GICv2 compatibility might choose to make this bit RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

## Accessing the ICC\_SRE\_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>	op1	CRn	CRm	op2
ICC_SRE_EL3	110	1100	1100	101

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	-	n/a	RW
0	1	-	-	-	RW
1	1	-	n/a	-	RW

This table applies to all instructions that can access this register.

This register is always System register accessible.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICH\_AP0R<n>\_EL2, Interrupt Controller Hyp Active Priorities Group 0 Registers, n = 0 - 3

The ICH\_AP0R<n>\_EL2 characteristics are:

## Purpose

Provides information about Group 0 virtual active priorities for EL2.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch64 System register ICH\_AP0R<n>\_EL2 is architecturally mapped to AArch32 System register [ICH\\_AP0R<n>](#).

If EL2 is not implemented, this register is RES0 from EL3.

## Attributes

ICH\_AP0R<n>\_EL2 is a 32-bit register.

## Field descriptions

The ICH\_AP0R<n>\_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
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

**P<x>, bit [x], for x = 0 to 31**

Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are:

P<x>	Meaning
0	There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop.
1	There is a Group 0 interrupt active with this priority level which has not undergone priority drop.

The correspondence between priority levels and bits depends on the number of bits of priority that are implemented.

If 5 bits of priority are implemented (bits [7:3] of priority), then there are 32 priority levels, and the active state of these priority levels are held in ICH\_AP0R0\_EL2 in the bits corresponding to Priority[7:3].

If 6 bits of priority are implemented (bits [7:2] of priority), then there are 64 priority levels, and:

- The active state of priority levels 0 - 124 are held in ICH\_AP0R0\_EL2 in the bits corresponding to 0:Priority[6:2].
- The active state of priority levels 128 - 252 are held in ICH\_AP0R1\_EL2 in the bits corresponding to 1:Priority[6:2].

If 7 bits of priority are implemented (bits [7:1] of priority), then there are 128 priority levels, and:

- The active state of priority levels 0 - 62 are held in ICH\_AP0R0\_EL2 in the bits corresponding to 00:Priority[5:1].
- The active state of priority levels 64 - 126 are held in ICH\_AP0R1\_EL2 in the bits corresponding to 01:Priority[5:1].
- The active state of priority levels 128 - 190 are held in ICH\_AP0R2\_EL2 in the bits corresponding to 10:Priority[5:1].
- The active state of priority levels 192 - 254 are held in ICH\_AP0R3\_EL2 in the bits corresponding to 11:Priority[5:1].

### Note



Having the bit corresponding to a priority set to 1 in both ICH\_AP0R<n>\_EL2 and [ICH\\_AP1R<n>\\_EL2](#) might result in UNPREDICTABLE behavior of the interrupt prioritization system for virtual interrupts.

When this register has an architecturally-defined reset value, this field resets to 0.

Software must ensure that ICH\_AP0R<n>\_EL2 is 0 for legacy VMs otherwise behaviour is UNPREDICTABLE. For more information about support for legacy VMs, see [Support for legacy operation of VMs](#).

The active priorities for Group 0 and Group 1 interrupts for legacy VMs are held in [ICH\\_AP1R<n>\\_EL2](#) and reads and writes to GICV\_APR access [ICH\\_AP1R<n>\\_EL2](#). This means that ICH\_AP0R<n>\_EL2 is inaccessible to legacy VMs.

## Accessing the ICH\_AP0R<n>\_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>	op1	CRn	CRm	op2
ICH_AP0R<n>_EL2	100	1100	1000	0:n<1:0>

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	-	n/a	RW
x	x	0	1	-	-	RW	RW
x	x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

ICH\_AP0R1\_EL2 is only implemented in implementations that support 6 or more bits of priority. ICH\_AP0R2\_EL2 and ICH\_AP0R3\_EL2 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

Writing to these registers with any value other than the last read value of the register (or 0x00000000 for a newly set up virtual machine) can result in UNPREDICTABLE behavior of the virtual interrupt prioritization system allowing either:

- Virtual interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution at Non-secure EL1 or EL0.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- ICH\_AP0R<n>\_EL2.
- [ICH\\_AP1R<n>\\_EL2](#).

Having the bit corresponding to a priority set in both ICH\_AP0R<n>\_EL2 and [ICH\\_AP1R<n>\\_EL2](#) can result in UNPREDICTABLE behavior of the interrupt prioritization system for virtual interrupts.

## 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 [ICC\\_SRE\\_EL2](#).SRE==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICH\_AP1R<n>\_EL2, Interrupt Controller Hyp Active Priorities Group 1 Registers, n = 0 - 3

The ICH\_AP1R<n>\_EL2 characteristics are:

## Purpose

Provides information about Group 1 virtual active priorities for EL2.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch64 System register ICH\_AP1R<n>\_EL2 is architecturally mapped to AArch32 System register [ICH\\_AP1R<n>](#).

If EL2 is not implemented, this register is RES0 from EL3.

## Attributes

ICH\_AP1R<n>\_EL2 is a 32-bit register.

## Field descriptions

The ICH\_AP1R<n>\_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
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

**P<x>, bit [x], for x = 0 to 31**

Group 1 interrupt active priorities. Possible values of each bit are:

P<x>	Meaning
0	There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop.
1	There is a Group 1 interrupt active with this priority level which has not undergone priority drop.

The correspondence between priority levels and bits depends on the number of bits of priority that are implemented.

If 5 bits of priority are implemented (bits [7:3] of priority), then there are 32 priority levels, and the active state of these priority levels are held in ICH\_AP1R0\_EL2 in the bits corresponding to Priority[7:3].

If 6 bits of priority are implemented (bits [7:2] of priority), then there are 64 priority levels, and:

- The active state of priority levels 0 - 124 are held in ICH\_AP1R0\_EL2 in the bits corresponding to 0:Priority[6:2].
- The active state of priority levels 128 - 252 are held in ICH\_AP1R1\_EL2 in the bits corresponding to 1:Priority[6:2].

If 7 bits of priority are implemented (bits [7:1] of priority), then there are 128 priority levels, and:

- The active state of priority levels 0 - 62 are held in ICH\_AP1R0\_EL2 in the bits corresponding to 00:Priority[5:1].
- The active state of priority levels 64 - 126 are held in ICH\_AP1R1\_EL2 in the bits corresponding to 01:Priority[5:1].
- The active state of priority levels 128 - 190 are held in ICH\_AP1R2\_EL2 in the bits corresponding to 10:Priority[5:1].
- The active state of priority levels 192 - 254 are held in ICH\_AP1R3\_EL2 in the bits corresponding to 11:Priority[5:1].

### Note

Having the bit corresponding to a priority set to 1 in both [ICH\\_AP0R<n>\\_EL2](#) and [ICH\\_AP1R<n>\\_EL2](#) might result in UNPREDICTABLE behavior of the interrupt prioritization system for virtual interrupts.

When this register has an architecturally-defined reset value, this field resets to 0.

This register is always used for legacy VMs, regardless of the group of the virtual interrupt. Reads and writes to [GICV\\_APR<n>](#) access [ICH\\_AP1R<n>\\_EL2](#). For more information about support for legacy VMs, see [Support for legacy operation of VMs](#).

## Accessing the ICH\_AP1R<n>\_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>	op1	CRn	CRm	op2
ICH_AP1R<n>_EL2	100	1100	1001	0:n<1:0>

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	-	n/a	RW
x	x	0	1	-	-	RW	RW
x	x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

ICH\_AP1R1\_EL2 is only implemented in implementations that support 6 or more bits of priority. ICH\_AP1R2\_EL2 and ICH\_AP1R3\_EL2 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

Writing to these registers with any value other than the last read value of the register (or 0x00000000 for a newly set up virtual machine) can result in UNPREDICTABLE behavior of the virtual interrupt prioritization system allowing either:

- Virtual interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution at Non-secure EL1 or EL0.

Writing to the active priority registers in any order other than the following order will result in UNPREDICTABLE behavior:

- [ICH\\_AP0R<n>\\_EL2](#).
- [ICH\\_AP1R<n>\\_EL2](#).

## 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 [ICC\\_SRE\\_EL2](#).SRE==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.



# ICH\_EISR\_EL2, Interrupt Controller End of Interrupt Status Register

The ICH\_EISR\_EL2 characteristics are:

## Purpose

Indicates which List registers have outstanding EOI maintenance interrupts.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch64 System register ICH\_EISR\_EL2 is architecturally mapped to AArch32 System register [ICH\\_EISR](#).

If EL2 is not implemented, this register is RES0 from EL3.

## Attributes

ICH\_EISR\_EL2 is a 32-bit register.

## Field descriptions

The ICH\_EISR\_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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Status<n>, bit [n], for n = 0 to 15															

### Bits [31:16]

Reserved, RES0.

### Status<n>, bit [n], for n = 0 to 15

EOI maintenance interrupt status bit for List register <n>:

Status<n>	Meaning
0	List register <n>, <a href="#">ICH_LR&lt;n&gt;_EL2</a> , does not have an EOI maintenance interrupt.
1	List register <n>, <a href="#">ICH_LR&lt;n&gt;_EL2</a> , has an EOI maintenance interrupt that has not been handled.

For any [ICH\\_LR<n>\\_EL2](#), the corresponding status bit is set to 1 if all of the following are true:

- [ICH\\_LR<n>\\_EL2](#).State is 0b00.
- [ICH\\_LR<n>\\_EL2](#).HW is 0.
- [ICH\\_LR<n>\\_EL2](#).EOI (bit [41]) is 1, indicating that when the interrupt corresponding to that List register is deactivated, a maintenance interrupt is asserted.

Otherwise the status bit takes the value 0.

## Accessing the ICH\_EISR\_EL2

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>	op1	CRn	CRm	op2
ICH_EISR_EL2	100	1100	1011	011

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	-	n/a	RO
0	1	-	-	RO	RO
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.

In both Security states, and not dependent on other configuration bits:

- If [ICC\\_SRE\\_EL2](#).SRE==0, read accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, read accesses to this register from EL3 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICH\_ELRSR\_EL2, Interrupt Controller Empty List Register Status Register

The ICH\_ELRSR\_EL2 characteristics are:

## Purpose

These registers can be used to locate a usable List register when the hypervisor is delivering an interrupt to a VM.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch64 System register ICH\_ELRSR\_EL2 is architecturally mapped to AArch32 System register [ICH\\_ELRSR](#).

If EL2 is not implemented, this register is RES0 from EL3.

## Attributes

ICH\_ELRSR\_EL2 is a 32-bit register.

## Field descriptions

The ICH\_ELRSR\_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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Status<n>, bit [n], for n = 0 to 15															

### Bits [31:16]

Reserved, RES0.

### Status<n>, bit [n], for n = 0 to 15

Status bit for List register <n>, [ICH\\_LR<n>\\_EL2](#):

Status<n>	Meaning
0	List register <a href="#">ICH_LR&lt;n&gt;_EL2</a> , if implemented, contains a valid interrupt. Using this List register can result in overwriting a valid interrupt.
1	List register <a href="#">ICH_LR&lt;n&gt;_EL2</a> does not contain a valid interrupt. The List register is empty and can be used without overwriting a valid interrupt or losing an EOI maintenance interrupt.

For any List register <n>, the corresponding status bit is set to 1 if [ICH\\_LR<n>\\_EL2](#).State is 0b00 and either [ICH\\_LR<n>\\_EL2](#).HW is 1 or [ICH\\_LR<n>\\_EL2](#).EOI (bit [41]) is 0.

Otherwise the status bit takes the value 0.



## Accessing the ICH\_ELRSR\_EL2

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>	op1	CRn	CRm	op2
ICH_ELRSR_EL2	100	1100	1011	101

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	-	n/a	RO
0	1	-	-	RO	RO
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.

In both Security states, and not dependent on other configuration bits:

- If [ICC\\_SRE\\_EL2](#).SRE==0, read accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, read accesses to this register from EL3 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICH\_HCR\_EL2, Interrupt Controller Hyp Control Register

The ICH\_HCR\_EL2 characteristics are:

## Purpose

Controls the environment for VMs.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch64 System register ICH\_HCR\_EL2 is architecturally mapped to AArch32 System register [ICH\\_HCR](#).

If EL2 is not implemented, this register is RES0 from EL3.

## Attributes

ICH\_HCR\_EL2 is a 32-bit register.

## Field descriptions

The ICH\_HCR\_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								
EOIcount																	TDIR										TSEI	TALL1	TALL0	TC	00	VGrp1DIE	VGrp1EIE	VGrp0DIE	VGrp0EIE	NPIEL	RENPIEL	UIE	En

### EOIcount, bits [31:27]

This field is incremented whenever a successful write to a virtual EOIR or DIR register would have resulted in a virtual interrupt deactivation. That is either:

- A virtual write to EOIR with a valid interrupt identifier that is not in the LPI range (i.e. < 8192) when EOI mode is zero and no List Register was found.
- A virtual write to DIR with a valid interrupt identifier that is not in the LPI range (i.e. < 8192) when EOI mode is one and no List Register was found.

This allows software to manage more active interrupts than there are implemented List Registers.

It is CONSTRAINED UNPREDICTABLE whether a virtual write to EOIR that does not clear a bit in the Active Priorities registers ([ICH\\_APOR<n>\\_EL2](#)/[ICH\\_APIR<n>\\_EL2](#)) increments EOIcount. Permitted behaviors are:

- Increment EOIcount.
- Leave EOIcount unchanged.

When this register has an architecturally-defined reset value, this field resets to 0.

### Bits [26:15]

Reserved, RES0.

### TDIR, bit [14]

Trap Non-secure EL1 writes to [ICC\\_DIR\\_EL1](#) and [ICV\\_DIR\\_EL1](#).

TDIR	Meaning
0	Non-secure EL1 writes of <a href="#">ICC_DIR_EL1</a> and <a href="#">ICV_DIR_EL1</a> are not trapped to EL2, unless trapped by other mechanisms.
1	Non-secure EL1 writes of <a href="#">ICC_DIR_EL1</a> and <a href="#">ICV_DIR_EL1</a> are trapped to EL2.

Support for this bit is OPTIONAL, with support indicated by [ICH\\_VTR\\_EL2](#).

If the implementation does not support this trap, this bit is RES0.

ARM deprecates not including this trap bit.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

### TSEI, bit [13]

Trap all locally generated SEIs. This bit allows the hypervisor to intercept locally generated SEIs that would otherwise be taken at Non-secure EL1.

TSEI	Meaning
0	Locally generated SEIs do not cause a trap to EL2.
1	Locally generated SEIs trap to EL2.

If [ICH\\_VTR\\_EL2](#).SEIS is 0, 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.

### TALL1, bit [12]

Trap all Non-secure EL1 accesses to ICC\_\* and ICV\_\* System registers for Group 1 interrupts to EL2.

TALL1	Meaning
0	Non-Secure EL1 accesses to ICC_* and ICV_* registers for Group 1 interrupts proceed as normal.
1	Non-secure EL1 accesses to ICC_* and ICV_* registers for Group 1 interrupts trap to EL2.

When this register has an architecturally-defined reset value, this field resets to 0.

### TALL0, bit [11]

Trap all Non-secure EL1 accesses to ICC\_\* and ICV\_\* System registers for Group 0 interrupts to EL2.

TALL0	Meaning
0	Non-Secure EL1 accesses to ICC_* and ICV_* registers for Group 0 interrupts proceed as normal.
1	Non-secure EL1 accesses to ICC_* and ICV_* registers for Group 0 interrupts trap to EL2.

When this register has an architecturally-defined reset value, this field resets to 0.

### TC, bit [10]

Trap all Non-secure EL1 accesses to System registers that are common to Group 0 and Group 1 to EL2.

TC	Meaning
0	Non-secure EL1 accesses to common registers proceed as normal.
1	Non-secure EL1 accesses to common registers trap to EL2.

This affects accesses to [ICC\\_SGI0R\\_EL1](#), [ICC\\_SGI1R\\_EL1](#), [ICC\\_ASGI1R\\_EL1](#), [ICC\\_CTLR\\_EL1](#), [ICC\\_DIR\\_EL1](#), [ICC\\_PMR\\_EL1](#), [ICC\\_RPR\\_EL1](#), [ICV\\_SGI0R\\_EL1](#), [ICV\\_SGI1R\\_EL1](#), [ICV\\_ASGI1R\\_EL1](#), [ICV\\_CTLR\\_EL1](#), [ICV\\_DIR\\_EL1](#), [ICV\\_PMR\\_EL1](#), and [ICV\\_RPR\\_EL1](#).

When this register has an architecturally-defined reset value, this field resets to 0.

**Bits [9:8]**

Reserved, RES0.

**VGrp1DIE, bit [7]**

VM Group 1 Disabled Interrupt Enable. Enables the signaling of a maintenance interrupt while signaling of Group 1 interrupts from the virtual CPU interface to the connected vPE is disabled:

VGrp1DIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled when <a href="#">ICH_VMCR_EL2.VENG1</a> is 0.

When this register has an architecturally-defined reset value, this field resets to 0.

**VGrp1EIE, bit [6]**

VM Group 1 Enabled Interrupt Enable. Enables the signaling of a maintenance interrupt while signaling of Group 1 interrupts from the virtual CPU interface to the connected vPE is enabled:

VGrp1EIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled when <a href="#">ICH_VMCR_EL2.VENG1</a> is 1.

When this register has an architecturally-defined reset value, this field resets to 0.

**VGrp0DIE, bit [5]**

VM Group 0 Disabled Interrupt Enable. Enables the signaling of a maintenance interrupt while signaling of Group 0 interrupts from the virtual CPU interface to the connected vPE is disabled:

VGrp0DIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled when <a href="#">ICH_VMCR_EL2.VENG0</a> is 0.

When this register has an architecturally-defined reset value, this field resets to 0.

**VGrp0EIE, bit [4]**

VM Group 0 Enabled Interrupt Enable. Enables the signaling of a maintenance interrupt while signaling of Group 0 interrupts from the virtual CPU interface to the connected vPE is enabled:

VGrp0EIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled when <a href="#">ICH_VMCR_EL2.VENG0</a> is 1.

When this register has an architecturally-defined reset value, this field resets to 0.

**NPiE, bit [3]**

No Pending Interrupt Enable. Enables the signaling of a maintenance interrupt while no pending interrupts are present in the List registers:

NPiE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled while the List registers contain no interrupts in the pending state.

When this register has an architecturally-defined reset value, this field resets to 0.

**LRNPiE, bit [2]**

List Register Entry Not Present Interrupt Enable. Enables the signaling of a maintenance interrupt while the virtual CPU interface does not have a corresponding valid List register entry for an EOI request:

<b>LRENPIE</b>	<b>Meaning</b>
0	Maintenance interrupt disabled.
1	Maintenance interrupt is asserted while the EOICount field is not 0.

When this register has an architecturally-defined reset value, this field resets to 0.

### UIE, bit [1]

Underflow Interrupt Enable. Enables the signaling of a maintenance interrupt when the List registers are empty, or hold only one valid entry:

<b>UIE</b>	<b>Meaning</b>
0	Maintenance interrupt disabled.
1	Maintenance interrupt is asserted if none, or only one, of the List register entries is marked as a valid interrupt.

When this register has an architecturally-defined reset value, this field resets to 0.

### En, bit [0]

Enable. Global enable bit for the virtual CPU interface:

<b>En</b>	<b>Meaning</b>
0	Virtual CPU interface operation disabled.
1	Virtual CPU interface operation enabled.

When this field is set to 0:

- The virtual CPU interface does not signal any maintenance interrupts.
- The virtual CPU interface does not signal any virtual interrupts.
- A read of [ICV\\_IAR0\\_EL1](#), [ICV\\_IAR1\\_EL1](#), [GICV\\_IAR](#) or [GICV\\_AIAR](#) returns a spurious interrupt ID.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the ICH\_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:

<b>&lt;systemreg&gt;</b>	<b>op1</b>	<b>CRn</b>	<b>CRm</b>	<b>op2</b>
ICH_HCR_EL2	100	1100	1011	000

## Accessibility

The register is accessible as follows:

<b>Control</b>		<b>Accessibility</b>			
<b>TGE</b>	<b>NS</b>	<b>EL0</b>	<b>EL1</b>	<b>EL2</b>	<b>EL3</b>
x	0	-	-	n/a	RW
0	1	-	-	RW	RW
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 [ICC\\_SRE\\_EL2](#).SRE==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICH\_LR<n>\_EL2, Interrupt Controller List Registers, n = 0 - 15

The ICH\_LR<n>\_EL2 characteristics are:

## Purpose

Provides interrupt context information for the virtual CPU interface.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch64 System register ICH\_LR<n>\_EL2 bits [31:0] are architecturally mapped to AArch32 System register [ICH\\_LR<n>](#).

AArch64 System register ICH\_LR<n>\_EL2 bits [63:32] are architecturally mapped to AArch32 System register [ICH\\_LRC<n>](#).

If EL2 is not implemented, this register is RES0 from EL3.

## Attributes

ICH\_LR<n>\_EL2 is a 64-bit register.

## Field descriptions

The ICH\_LR<n>\_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
State		HWGroup		0	0	0	0	Priority								0	0	0	0	0	0	pINTID									
vINTID																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### State, bits [63:62]

The state of the interrupt:

State	Meaning
00	Inactive
01	Pending
10	Active
11	Pending and active.

The GIC updates these state bits as virtual interrupts proceed through the interrupt life cycle. Entries in the inactive state are ignored, except for the purpose of generating virtual maintenance interrupts.

For hardware interrupts, the pending and active state is held in the physical Distributor rather than the virtual CPU interface. A hypervisor must only use the pending and active state for software originated interrupts, which are typically associated with virtual devices, or SGIs.

When this register has an architecturally-defined reset value, this field resets to 0.

### HW, bit [61]

Indicates whether this virtual interrupt maps directly to a hardware interrupt, meaning that it corresponds to a physical interrupt. Deactivation of the virtual interrupt also causes the deactivation of the physical interrupt with the ID that the pINTID field indicates.

HW	Meaning
0	The interrupt is triggered entirely by software. No notification is sent to the Distributor when the virtual interrupt is deactivated.
1	The interrupt maps directly to a hardware interrupt. A deactivate interrupt request is sent to the Distributor when the virtual interrupt is deactivated, using the pINTID field from this register to indicate the physical interrupt ID. If <a href="#">ICH_VMCR_EL2.VEOIM</a> is 0, this request corresponds to a write to <a href="#">ICC_EOIR0_EL1</a> or <a href="#">ICC_EOIR1_EL1</a> . Otherwise, it corresponds to a write to <a href="#">ICC_DIR_EL1</a> .

When this register has an architecturally-defined reset value, this field resets to 0.

#### Group, bit [60]

Indicates the group for this virtual interrupt.

Group	Meaning
0	This is a Group 0 virtual interrupt. <a href="#">ICH_VMCR_EL2.VFIQEn</a> determines whether it is signaled as a virtual IRQ or as a virtual FIQ, and <a href="#">ICH_VMCR_EL2.VENG0</a> enables signaling of this interrupt to the virtual machine.
1	This is a Group 1 virtual interrupt, signaled as a virtual IRQ. <a href="#">ICH_VMCR_EL2.VENG1</a> enables the signaling of this interrupt to the virtual machine. If <a href="#">ICH_VMCR_EL2.VCBPR</a> is 0, then <a href="#">ICC_BPR1_EL1</a> determines if a pending Group 1 interrupt has sufficient priority to preempt current execution. Otherwise, <a href="#">ICH_LR&lt;n&gt;_EL2</a> determines preemption.

When this register has an architecturally-defined reset value, this field resets to 0.

#### Bits [59:56]

Reserved, RES0.

#### Priority, bits [55:48]

The priority of this interrupt.

It is IMPLEMENTATION DEFINED how many bits of priority are implemented, though at least five bits must be implemented. Unimplemented bits are RES0 and start from bit [48] up to bit [50]. The number of implemented bits can be discovered from [ICH\\_VTR\\_EL2.PR](#)bits.

When this register has an architecturally-defined reset value, this field resets to 0.

#### Bits [47:42]

Reserved, RES0.

#### pINTID, bits [41:32]

Physical INTID, for hardware interrupts.

When the HW bit is 0 (there is no corresponding physical interrupt), this field has the following meaning:

- Bit [41] : EOI. If this bit is 1, then when the interrupt identified by vINTID is deactivated, a maintenance interrupt is asserted.
- Bits [40:32] : Reserved, RES0.

When the HW bit is 1 (there is a corresponding physical interrupt):

- This field indicates the physical INTID. This field is only required to implement enough bits to hold a valid value for the implemented INTID size. Any unused higher order bits are RES0.
- If the value of pINTID is 0-15 or 1020-1023, behavior is UNPREDICTABLE. If the value of pINTID is 16-31, this field applies to the PPI associated with this same physical PE ID as the virtual CPU interface requesting the deactivation.

A hardware physical identifier is only required in List Registers for interrupts that require deactivation. This means only 10 bits of Physical INTID are required, regardless of the number specified by [ICC\\_CTLR\\_EL1.ID](#)bits.



When this register has an architecturally-defined reset value, this field resets to 0.

## vINTID, bits [31:0]

Virtual INTID of the interrupt.

Behavior is UNPREDICTABLE if two or more List Registers specify the same vINTID when:

- ICH\_LR<n>\_EL2.State == 01.
- ICH\_LR<n>\_EL2.State == 10.
- ICH\_LR<n>\_EL2.State == 11.

It is IMPLEMENTATION DEFINED how many bits are implemented, though at least 16 bits must be implemented. Unimplemented bits are RES0. The number of implemented bits can be discovered from [ICH\\_VTR\\_EL2.IDbits](#).

### Note

When a VM is using memory-mapped access to the GIC, software must ensure that the correct source PE ID is provided in bits[12:10].

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the ICH\_LR<n>\_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>	op1	CRn	CRm	op2
ICH_LR<n>_EL2	100	1100	110:n<3>	n<2:0>

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	-	n/a	RW
0	1	-	-	RW	RW
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 [ICC\\_SRE\\_EL2.SRE](#)==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3.SRE](#)==0, accesses to this register from EL3 are trapped to EL3.



# ICH\_MISR\_EL2, Interrupt Controller Maintenance Interrupt State Register

The ICH\_MISR\_EL2 characteristics are:

## Purpose

Indicates which maintenance interrupts are asserted.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch64 System register ICH\_MISR\_EL2 is architecturally mapped to AArch32 System register [ICH\\_MISR](#).

If EL2 is not implemented, this register is RES0 from EL3.

## Attributes

ICH\_MISR\_EL2 is a 32-bit register.

## Field descriptions

The ICH\_MISR\_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	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	VGrp1D	VGrp1E	VGrp0D	VGrp0E	NPL	REN	P	U	EOI

### Bits [31:8]

Reserved, RES0.

### VGrp1D, bit [7]

vPE Group 1 Disabled.

VGrp1D	Meaning
0	vPE Group 1 Disabled maintenance interrupt not asserted.
1	vPE Group 1 Disabled maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR\\_EL2](#).VGrp1DIE==1 and [ICH\\_VMCR\\_EL2](#).VENG1==is 0.

When this register has an architecturally-defined reset value, this field resets to 0.

### VGrp1E, bit [6]

vPE Group 1 Enabled.

VGrp1E	Meaning
0	vPE Group 1 Enabled maintenance interrupt not asserted.
1	vPE Group 1 Enabled maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR\\_EL2](#).VGrp1EIE==1 and [ICH\\_VMCR\\_EL2](#).VENG1==is 1.

When this register has an architecturally-defined reset value, this field resets to 0.

#### VGrp0D, bit [5]

vPE Group 0 Disabled.

VGrp0D	Meaning
0	vPE Group 0 Disabled maintenance interrupt not asserted.
1	vPE Group 0 Disabled maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR\\_EL2.VGrp0DIE](#)==1 and [ICH\\_VMCR\\_EL2.VENG0](#)==0.

When this register has an architecturally-defined reset value, this field resets to 0.

#### VGrp0E, bit [4]

vPE Group 0 Enabled.

VGrp0E	Meaning
0	vPE Group 0 Enabled maintenance interrupt not asserted.
1	vPE Group 0 Enabled maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR\\_EL2.VGrp0EIE](#)==1 and [ICH\\_VMCR\\_EL2.VENG0](#)==1.

When this register has an architecturally-defined reset value, this field resets to 0.

#### NP, bit [3]

No Pending.

NP	Meaning
0	No Pending maintenance interrupt not asserted.
1	No Pending maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR\\_EL2.NPIE](#)==1 and no List register is in pending state.

When this register has an architecturally-defined reset value, this field resets to 0.

#### LRENP, bit [2]

List Register Entry Not Present.

LRENP	Meaning
0	List Register Entry Not Present maintenance interrupt not asserted.
1	List Register Entry Not Present maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR\\_EL2.LRENPIE](#)==1 and [ICH\\_HCR\\_EL2.EOIcount](#) is non-zero.

When this register has an architecturally-defined reset value, this field resets to 0.

#### U, bit [1]

Underflow.

U	Meaning
0	Underflow maintenance interrupt not asserted.
1	Underflow maintenance interrupt asserted.

This maintenance interrupt is asserted when [ICH\\_HCR\\_EL2.UIE](#)==1 and zero or one of the List register entries are marked as a valid interrupt, that is, if the corresponding [ICH\\_LR<n>\\_EL2.State](#) bits do not equal 0x0.

When this register has an architecturally-defined reset value, this field resets to 0.

## EOI, bit [0]

End Of Interrupt.

EOI	Meaning
0	End Of Interrupt maintenance interrupt not asserted.
1	End Of Interrupt maintenance interrupt asserted.

This maintenance interrupt is asserted when at least one bit in [ICH\\_EISR\\_EL2](#) is 1.

When this register has an architecturally-defined reset value, this field resets to 0.

The U and NP bits do not include the status of any pending/active VSET packets because these bits control generation of interrupts that allow software management of the contents of the List Registers (which are not affected by VSET packets).

## Accessing the ICH\_MISR\_EL2

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>	op1	CRn	CRm	op2
ICH_MISR_EL2	100	1100	1011	010

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	-	n/a	RO
0	1	-	-	RO	RO
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.

In both Security states, and not dependent on other configuration bits:

- If [ICC\\_SRE\\_EL2](#).SRE==0, read accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, read accesses to this register from EL3 are trapped to EL3.

# ICH\_VMCR\_EL2, Interrupt Controller Virtual Machine Control Register

The ICH\_VMCR\_EL2 characteristics are:

## Purpose

Enables the hypervisor to save and restore the virtual machine view of the GIC state.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch64 System register ICH\_VMCR\_EL2 is architecturally mapped to AArch32 System register [ICH\\_VMCR](#).

If EL2 is not implemented, this register is RES0 from EL3.

## Attributes

ICH\_VMCR\_EL2 is a 32-bit register.

## Field descriptions

The ICH\_VMCR\_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
VPMR								VBPR0	VBPR1	0	0	0	0	0	0	0	0	0	0	0	0	VEOIM	0	0	0	0	VCBPR	VFIQEn	VAckCtl	VENG1	VENG0

### VPMR, bits [31:24]

Virtual Priority Mask. The priority mask level for the virtual CPU interface. If the priority of a pending virtual interrupt is higher than the value indicated by this field, the interface signals the virtual interrupt to the PE.

This field is an alias of [ICV\\_PMR\\_EL1](#).Priority.

### VBPR0, bits [23:21]

Virtual Binary Point Register, Group 0. Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 0 interrupt preemption, and also determines Group 1 interrupt preemption if  $ICH\_VMCR\_EL2.VCBPR == 1$ .

This field is an alias of [ICV\\_BPR0\\_EL1](#).BinaryPoint.

The minimum value of this field is determined by [ICH\\_VTR\\_EL2](#).PREbits. An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value.

### VBPR1, bits [20:18]

Virtual Binary Point Register, Group 1. Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption if  $ICH\_VMCR\_EL2.VCBPR == 0$ .

This field is an alias of [ICV\\_BPR1\\_EL1](#).BinaryPoint.

**Bits [17:10]**

Reserved, RES0.

**VEOIM, bit [9]**

Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt:

VEOIM	Meaning
0	<a href="#">ICV_EOIR0_EL1</a> and <a href="#">ICV_EOIR1_EL1</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">ICV_DIR_EL1</a> are UNPREDICTABLE.
1	<a href="#">ICV_EOIR0_EL1</a> and <a href="#">ICV_EOIR1_EL1</a> provide priority drop functionality only. <a href="#">ICV_DIR_EL1</a> provides interrupt deactivation functionality.

This bit is an alias of [ICV\\_CTLR\\_EL1](#).EOImode.

**Bits [8:5]**

Reserved, RES0.

**VCBPR, bit [4]**

Virtual Common Binary Point Register. Possible values of this bit are:

VCBPR	Meaning
0	<a href="#">ICV_BPR0_EL1</a> determines the preemption group for virtual Group 0 interrupts only.
1	<a href="#">ICV_BPR1_EL1</a> determines the preemption group for virtual Group 1 interrupts. <a href="#">ICV_BPR0_EL1</a> determines the preemption group for both virtual Group 0 and virtual Group 1 interrupts. Reads of <a href="#">ICV_BPR1_EL1</a> return <a href="#">ICV_BPR0_EL1</a> plus one, saturated to 0b111. Writes to <a href="#">ICV_BPR1_EL1</a> are ignored.

This field is an alias of [ICV\\_CTLR\\_EL1](#).CBPR.

**VFIQEn, bit [3]**

Virtual FIQ enable. Possible values of this bit are:

VFIQEn	Meaning
0	Group 0 virtual interrupts are presented as virtual IRQs.
1	Group 0 virtual interrupts are presented as virtual FIQs.

This bit is an alias of [GICV\\_CTLR](#).FIQEn.

In implementations where the Non-secure copy of [ICC\\_SRE\\_EL1](#).SRE is always 1, this bit is RES1.

**VAckCtl, bit [2]**

Virtual AckCtl. Possible values of this bit are:

VAckCtl	Meaning
0	If the highest priority pending interrupt is Group 1, a read of <a href="#">GICV_IAR</a> or <a href="#">GICV_HPPIR</a> returns an INTID of 1022.
1	If the highest priority pending interrupt is Group 1, a read of <a href="#">GICV_IAR</a> or <a href="#">GICV_HPPIR</a> returns the INTID of the corresponding interrupt.

This bit is an alias of [GICV\\_CTLR](#).AckCtl.

This field is supported for backwards compatibility with GICv2. ARM deprecates the use of this field.

In implementations where the Non-secure copy of [ICC\\_SRE\\_EL1](#).SRE is always 1, this bit is RES0.

## VENG1, bit [1]

Virtual Group 1 interrupt enable. Possible values of this bit are:

VENG1	Meaning
0	Virtual Group 1 interrupts are disabled.
1	Virtual Group 1 interrupts are enabled.

This bit is an alias of [ICV\\_IGRPEN1\\_EL1](#). Enable.

## VENG0, bit [0]

Virtual Group 0 interrupt enable. Possible values of this bit are:

VENG0	Meaning
0	Virtual Group 0 interrupts are disabled.
1	Virtual Group 0 interrupts are enabled.

This bit is an alias of [ICV\\_IGRPEN0\\_EL1](#). Enable.

# Accessing the ICH\_VMCR\_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>	op1	CRn	CRm	op2
ICH_VMCR_EL2	100	1100	1011	111

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	-	n/a	RW
0	1	-	-	RW	RW
1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

When EL2 is using System register access, EL1 using either System register or memory-mapped access must be supported.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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 [ICC\\_SRE\\_EL2](#).SRE==0, accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, accesses to this register from EL3 are trapped to EL3.





# ICH\_VTR\_EL2, Interrupt Controller VGIC Type Register

The ICH\_VTR\_EL2 characteristics are:

## Purpose

Reports supported GIC virtualisartion features.

This register is part of:

- The GIC system registers functional group.
- The Virtualization registers functional group.
- The GIC host interface control registers functional group.

## Configuration

AArch64 System register ICH\_VTR\_EL2 is architecturally mapped to AArch32 System register [ICH\\_VTR](#).

If EL2 is not implemented, all bits in this register are RES0 from EL3, except for nV4, which is RES1 from EL3.

## Attributes

ICH\_VTR\_EL2 is a 32-bit register.

## Field descriptions

The ICH\_VTR\_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
<a href="#">PRibits</a>	<a href="#">PREbits</a>	<a href="#">IDbits</a>	<a href="#">SEIS</a>	<a href="#">A3V</a>	<a href="#">nV4</a>	<a href="#">TDS</a>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<a href="#">ListRegs</a>					

### PRibits, bits [31:29]

Priority bits. The number of virtual priority bits implemented, minus one.

An implementation must implement at least 32 levels of virtual priority (5 priority bits).

This field is an alias of [ICV\\_CTLR\\_EL1](#).PRibits.

### PREbits, bits [28:26]

The number of virtual preemption bits implemented, minus one.

An implementation must implement at least 32 levels of virtual preemption priority (5 preemption bits).

The value of this field must be less than or equal to the value of ICH\_VTR\_EL2.PRibits.

This field determines the minimum value of [ICH\\_VMCR\\_EL2](#).VBPR0.

### IDbits, bits [25:23]

The number of virtual interrupt identifier bits supported:

IDbits	Meaning
000	16 bits.
001	24 bits.

All other values are reserved.

This field is an alias of [ICV\\_CTLR\\_EL1](#).IDbits.

### SEIS, bit [22]

SEI Support. Indicates whether the virtual CPU interface supports generation of SEIs:

SEIS	Meaning
0	The virtual CPU interface logic does not support generation of SEIs.
1	The virtual CPU interface logic supports generation of SEIs.

This bit is an alias of [ICV\\_CTLR\\_EL1](#).SEIS.

### A3V, bit [21]

Affinity 3 Valid. Possible values are:

A3V	Meaning
0	The virtual CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers.
1	The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

This bit is an alias of [ICV\\_CTLR\\_EL1](#).A3V.

### nV4, bit [20]

Direct injection of virtual interrupts not supported. Possible values are:

nV4	Meaning
0	The CPU interface logic supports direct injection of virtual interrupts.
1	The CPU interface logic does not support direct injection of virtual interrupts.

In GICv3 this bit is RES1.

### TDS, bit [19]

Separate trapping of Non-secure EL1 writes to [ICV\\_DIR\\_EL1](#) supported.

TDS	Meaning
0	Implementation does not support <a href="#">ICV_HCR_EL2</a> .TDIR.
1	Implementation supports <a href="#">ICV_HCR_EL2</a> .TDIR.

### Bits [18:5]

Reserved, RES0.

### ListRegs, bits [4:0]

The number of implemented List registers, minus one. For example, a value of 0b01111 indicates that the maximum of 16 List registers are implemented.

## Accessing the ICH\_VTR\_EL2

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>	op1	CRn	CRm	op2
ICH_VTR_EL2	100	1100	1011	001

## Accessibility

The register is accessible as follows:

Control		Accessibility			
TGE	NS	EL0	EL1	EL2	EL3
x	0	-	-	n/a	RO
0	1	-	-	RO	RO
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<sup>®</sup> 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 [ICC\\_SRE\\_EL2](#).SRE==0, read accesses to this register from EL2 are trapped to EL2.
- If [ICC\\_SRE\\_EL3](#).SRE==0, read accesses to this register from EL3 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICV\_AP0R<n>\_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers, n = 0 - 3

The ICV\_AP0R<n>\_EL1 characteristics are:

## Purpose

Provides information about virtual Group 0 active priorities.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_AP0R<n>\_EL1 is architecturally mapped to AArch32 System register [ICV\\_AP0R<n>](#).

## Attributes

ICV\_AP0R<n>\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_AP0R<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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

When this register has an architecturally-defined reset value, this field resets to 0.

The contents of these registers are IMPLEMENTATION DEFINED with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

## Accessing the ICV\_AP0R<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>	op1	CRn	CRm	op2
ICC_AP0R<n>_EL1	000	1100	1000	1:n<1:0>

When HCR\_EL2.FMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_AP0R<n>\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_AP0R&lt;n&gt;_EL1</a>	n/a	<a href="#">ICC_AP0R&lt;n&gt;_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_AP0R&lt;n&gt;_EL1</a>	<a href="#">ICC_AP0R&lt;n&gt;_EL1</a>
0	x	0	1	-	<a href="#">ICC_AP0R&lt;n&gt;_EL1</a>	<a href="#">ICC_AP0R&lt;n&gt;_EL1</a>	<a href="#">ICC_AP0R&lt;n&gt;_EL1</a>
1	x	0	1	-	RW	<a href="#">ICC_AP0R&lt;n&gt;_EL1</a>	<a href="#">ICC_AP0R&lt;n&gt;_EL1</a>

This table applies to all instructions that can access this register.

The ICV\_AP0R<n>\_EL1 registers are only accessible at Non-secure EL1 when HCR\_EL2.FMO is set to 1.

### Note

When HCR\_EL2.FMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_AP0R<n>\_EL1 results in an access to [ICC\\_AP0R<n>\\_EL1](#).

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 0 active priorities) might result in UNPREDICTABLE behavior of the virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV\_AP0R1\_EL1 is only implemented in implementations that support 6 or more bits of priority. ICV\_AP0R2\_EL1 and ICV\_AP0R3\_EL1 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

Writing to the active priority registers in any order other than the following order might result in UNPREDICTABLE behavior of the interrupt prioritization system:

- ICV\_AP0R<n>\_EL1.
- [ICV\\_APIR<n>\\_EL1](#).

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure accesses to this register from EL1 are trapped to EL2.

# ICV\_AP1R<n>\_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers, n = 0 - 3

The ICV\_AP1R<n>\_EL1 characteristics are:

## Purpose

Provides information about virtual Group 1 active priorities.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_AP1R<n>\_EL1 is architecturally mapped to AArch32 System register [ICV\\_AP1R<n>](#).

## Attributes

ICV\_AP1R<n>\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_AP1R<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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

When this register has an architecturally-defined reset value, this field resets to 0.

The contents of these registers are IMPLEMENTATION DEFINED with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

## Accessing the ICV\_AP1R<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>	op1	CRn	CRm	op2
ICC_AP1R<n>_EL1	000	1100	1001	0:n<1:0>

When HCR\_EL2.IMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_AP1R<n>\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_AP1R&lt;n&gt;_EL1</a>	n/a	<a href="#">ICC_AP1R&lt;n&gt;_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_AP1R&lt;n&gt;_EL1</a>	<a href="#">ICC_AP1R&lt;n&gt;_EL1</a>
x	0	0	1	-	<a href="#">ICC_AP1R&lt;n&gt;_EL1</a>	<a href="#">ICC_AP1R&lt;n&gt;_EL1</a>	<a href="#">ICC_AP1R&lt;n&gt;_EL1</a>
x	1	0	1	-	RW	<a href="#">ICC_AP1R&lt;n&gt;_EL1</a>	<a href="#">ICC_AP1R&lt;n&gt;_EL1</a>

This table applies to all instructions that can access this register.

The ICV\_AP1R<n>\_EL1 registers are only accessible at Non-secure EL1 when HCR\_EL2.IMO == 1.

### Note

When HCR\_EL2.IMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_AP1R<n>\_EL1 results in an access to [ICC\\_AP1R<n>\\_EL1](#).

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 1 active priorities) might result in UNPREDICTABLE behavior of the virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV\_AP1R1\_EL1 is only implemented in implementations that support 6 or more bits of priority. ICV\_AP1R2\_EL1 and ICV\_AP1R3\_EL1 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are UNDEFINED.

Writing to the active priority registers in any order other than the following order might result in UNPREDICTABLE behavior of the interrupt prioritization system:

- [ICV\\_AP0R<n>\\_EL1](#).
- ICV\_AP1R<n>\_EL1.

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure accesses to this register from EL1 are trapped to EL2.



# ICV\_BPR0\_EL1, Interrupt Controller Virtual Binary Point Register 0

The ICV\_BPR0\_EL1 characteristics are:

## Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines virtual Group 0 interrupt preemption.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_BPR0\_EL1 is architecturally mapped to AArch32 System register [ICV\\_BPR0](#).

## Attributes

ICV\_BPR0\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_BPR0\_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	0	0	BinaryPoint

### Bits [31:3]

Reserved, RES0.

### BinaryPoint, bits [2:0]

The value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. This is done as follows:

Binary point value	Group priority field	Subpriority field	Field with binary point
0	[7:1]	[0]	ggggggg.s
1	[7:2]	[1:0]	ggggggg.ss
2	[7:3]	[2:0]	ggggggg.sss
3	[7:4]	[3:0]	ggggg.ssss
4	[7:5]	[4:0]	ggg.sssss
5	[7:6]	[5:0]	gg.ssssss
6	[7]	[6:0]	g.sssssss
7	No preemption	[7:0]	.ssssssss

## Accessing the ICV\_BPR0\_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>	op1	CRn	CRm	op2
ICC_BPR0_EL1	000	1100	1000	011

When HCR\_EL2.FMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_BPR0\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_BPR0_EL1</a>	n/a	<a href="#">ICC_BPR0_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_BPR0_EL1</a>	<a href="#">ICC_BPR0_EL1</a>
0	x	0	1	-	<a href="#">ICC_BPR0_EL1</a>	<a href="#">ICC_BPR0_EL1</a>	<a href="#">ICC_BPR0_EL1</a>
1	x	0	1	-	RW	<a href="#">ICC_BPR0_EL1</a>	<a href="#">ICC_BPR0_EL1</a>

This table applies to all instructions that can access this register.

ICV\_BPR0\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.FMO is set to 1.

---

### Note

When HCR\_EL2.FMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_BPR0\_EL1 results in an access to [ICC\\_BPR0\\_EL1](#).

---

The minimum binary point value is derived from the number of implemented priority bits. The number of priority bits is IMPLEMENTATION DEFINED, and reported by [ICV\\_CTLR\\_EL1](#).PRIBits.

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value. On a reset, the binary point field is set to the minimum supported value.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure accesses to this register from EL1 are trapped to EL2.

# ICV\_BPR1\_EL1, Interrupt Controller Virtual Binary Point Register 1

The ICV\_BPR1\_EL1 characteristics are:

## Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines virtual Group 1 interrupt preemption.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_BPR1\_EL1 is architecturally mapped to AArch32 System register [ICV\\_BPR1](#).

## Attributes

ICV\_BPR1\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_BPR1\_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	0	0	BinaryPoint

### Bits [31:3]

Reserved, RES0.

### BinaryPoint, bits [2:0]

If the GIC is configured to use separate binary point fields for virtual Group 0 and virtual Group 1 interrupts, the value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. This is done as follows:

Binary point value	Group priority field	Subpriority field	Field with binary point
0	-	-	-
1	[7:1]	[0]	ggggggg.s
2	[7:2]	[1:0]	gggggg.ss
3	[7:3]	[2:0]	ggggg.sss
4	[7:4]	[3:0]	gggg.ssss
5	[7:5]	[4:0]	ggg.sssss
6	[7:6]	[5:0]	gg.ssssss
7	[7]	[6:0]	g.sssssss

Writing 0 to this field will set this field to its reset value, which is IMPLEMENTATION DEFINED and non-zero.

If [ICV\\_CTLR\\_EL1](#).CBPR is set to 1, Non-secure EL1 reads return [ICV\\_BPR0\\_EL1](#) + 1 saturated to 0b111. Non-secure EL1 writes are ignored.

## Accessing the ICV\_BPR1\_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>	op1	CRn	CRm	op2
ICC_BPR1_EL1	000	1100	1100	011

When HCR\_EL2.IMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_BPR1\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_BPR1_EL1</a>	n/a	<a href="#">ICC_BPR1_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_BPR1_EL1</a>	<a href="#">ICC_BPR1_EL1</a>
x	0	0	1	-	<a href="#">ICC_BPR1_EL1</a>	<a href="#">ICC_BPR1_EL1</a>	<a href="#">ICC_BPR1_EL1</a>
x	1	0	1	-	RW	<a href="#">ICC_BPR1_EL1</a>	<a href="#">ICC_BPR1_EL1</a>

This table applies to all instructions that can access this register.

ICV\_BPR1\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.IMO is set to 1.

### Note

When HCR\_EL2.IMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_BPR1\_EL1 results in an access to [ICC\\_BPR1\\_EL1](#).

The reset value is IMPLEMENTATION DEFINED, but is equal to the minimum value of [ICV\\_BPR0\\_EL1](#) plus one.

An attempt to program the binary point field to a value less than the reset value sets the field to the reset value.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure accesses to this register from EL1 are trapped to EL2.

# ICV\_CTLR\_EL1, Interrupt Controller Virtual Control Register

The ICV\_CTLR\_EL1 characteristics are:

## Purpose

Controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_CTLR\_EL1 is architecturally mapped to AArch32 System register [ICV\\_CTLR](#).

## Attributes

ICV\_CTLR\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_CTLR\_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	<a href="#">A3V</a>	<a href="#">SEIS</a>	<a href="#">IDbits</a>	<a href="#">PRIbits</a>	0	0	0	0	0	0	0	0	<a href="#">EOImode</a>	<a href="#">CBPR</a>		

### Bits [31:16]

Reserved, RES0.

### A3V, bit [15]

Affinity 3 Valid. Read-only and writes are ignored. Possible values are:

A3V	Meaning
0	The virtual CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers.
1	The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.

### SEIS, bit [14]

SEI Support. Read-only and writes are ignored. Indicates whether the virtual CPU interface supports local generation of SEIs:

SEIS	Meaning
0	The virtual CPU interface logic does not support local generation of SEIs.
1	The virtual CPU interface logic supports local generation of SEIs.

### IDbits, bits [13:11]

Identifier bits. Read-only and writes are ignored. The number of virtual interrupt identifier bits supported:

IDbits	Meaning
000	16 bits.
001	24 bits.

All other values are reserved.

### PRibits, bits [10:8]

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation must implement at least 32 levels of physical priority (5 priority bits).

#### Note

This field always returns the number of priority bits implemented.

The division between group priority and subpriority is defined in the binary point registers [ICV\\_BPR0\\_EL1](#) and [ICV\\_BPR1\\_EL1](#).

### Bits [7:2]

Reserved, RES0.

### EOImode, bit [1]

Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt:

EOImode	Meaning
0	<a href="#">ICV_EOIR0_EL1</a> and <a href="#">ICV_EOIR1_EL1</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">ICV_DIR_EL1</a> are UNPREDICTABLE.
1	<a href="#">ICV_EOIR0_EL1</a> and <a href="#">ICV_EOIR1_EL1</a> provide priority drop functionality only. <a href="#">ICV_DIR_EL1</a> provides interrupt deactivation functionality.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### CBPR, bit [0]

Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts:

CBPR	Meaning
0	<a href="#">ICV_BPR0_EL1</a> determines the preemption group for virtual Group 0 interrupts only.
1	<a href="#">ICV_BPR1_EL1</a> determines the preemption group for virtual Group 1 interrupts. <a href="#">ICV_BPR0_EL1</a> determines the preemption group for both virtual Group 0 and virtual Group 1 interrupts. Reads of <a href="#">ICV_BPR1_EL1</a> return <a href="#">ICV_BPR0_EL1</a> plus one, saturated to 0b111. Writes to <a href="#">ICV_BPR1_EL1</a> are ignored.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the ICV\_CTLR\_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>	op1	CRn	CRm	op2
ICC_CTLR_EL1	000	1100	1100	100

When HCR\_EL2.{FMO, IMO} == {0, 0}, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_CTLR\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_CTLR_EL1</a>	n/a	<a href="#">ICC_CTLR_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_CTLR_EL1</a>	<a href="#">ICC_CTLR_EL1</a>
x	1	0	1	-	RW	<a href="#">ICC_CTLR_EL1</a>	<a href="#">ICC_CTLR_EL1</a>
1	x	0	1	-	RW	<a href="#">ICC_CTLR_EL1</a>	<a href="#">ICC_CTLR_EL1</a>
0	0	0	1	-	<a href="#">ICC_CTLR_EL1</a>	<a href="#">ICC_CTLR_EL1</a>	<a href="#">ICC_CTLR_EL1</a>

This table applies to all instructions that can access this register.

ICV\_CTLR\_EL1 is only accessible at Non-secure EL1 when  $\text{HCR\_EL2}\{FMO, IMO\} \neq \{0, 0\}$ .

---

### Note

When  $\text{HCR\_EL2}\{FMO, IMO\} = \{0, 0\}$ , at Non-secure EL1, the instruction encoding used to access ICV\_CTLR\_EL1 results in an access to [ICC\\_CTLR\\_EL1](#).

---

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure accesses to this register from EL1 are trapped to EL1.

When  $\text{SCR\_EL3.NS}=1$  :

- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ICV\_DIR\_EL1, Interrupt Controller Deactivate Virtual Interrupt Register

The ICV\_DIR\_EL1 characteristics are:

## Purpose

When interrupt priority drop is separated from interrupt deactivation, a write to this register deactivates the specified virtual interrupt.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_DIR\_EL1 performs the same function as AArch32 System register [ICV\\_DIR](#).

## Attributes

ICV\_DIR\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_DIR\_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																								

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the virtual interrupt to be deactivated.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICV\\_CTLR\\_EL1](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

## Accessing the ICV\_DIR\_EL1

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>	op1	CRn	CRm	op2
ICC_DIR_EL1	000	1100	1011	001

This encoding results in an access to ICV\_DIR\_EL1 at Non-secure EL1 in the following cases:

- When HCR\_EL2.FMO is set to 1.
- When HCR\_EL2.IMO is set to 1.

This encoding results in an access to [ICC\\_DIR\\_EL1](#) at Non-secure EL1 in the following cases:



- When  $\text{HCR\_EL2}\{FMO, IMO\} = \{0, 0\}$ .

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_DIR_EL1</a>	n/a	<a href="#">ICC_DIR_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_DIR_EL1</a>	<a href="#">ICC_DIR_EL1</a>
x	1	0	1	-	WO	<a href="#">ICC_DIR_EL1</a>	<a href="#">ICC_DIR_EL1</a>
1	x	0	1	-	WO	<a href="#">ICC_DIR_EL1</a>	<a href="#">ICC_DIR_EL1</a>
0	0	0	1	-	<a href="#">ICC_DIR_EL1</a>	<a href="#">ICC_DIR_EL1</a>	<a href="#">ICC_DIR_EL1</a>

This table applies to all instructions that can access this register.

The ICV\_DIR\_EL1 register is only accessible at Non-secure EL1 in the following cases:

- When  $\text{HCR\_EL2}.FMO$  is set to 1.
- When  $\text{HCR\_EL2}.IMO$  is set to 1.

---

### Note

At Non-secure EL1, the instruction encoding used to access ICV\_DIR\_EL1 results in an access to [ICC\\_DIR\\_EL1](#) when  $\text{HCR\_EL2}\{FMO, IMO\} = \{0, 0\}$ .

---

When  $\text{EOImode} = 0$ , writes are ignored. In systems supporting system error generation, an implementation might generate an SEI.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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  $\text{ICC\_SRE\_EL1}.SRE = 0$ , Non-secure write accesses to this register from EL1 are trapped to EL1.

When  $\text{SCR\_EL3}.NS = 1$ :

- If  $\text{ICH\_HCR\_EL2}.TC = 1$ , Non-secure write accesses to this register from EL1 are trapped to EL2.
- If  $\text{ICH\_HCR\_EL2}.TDIR = 1$ , Non-secure write accesses to this register from EL1 are trapped to EL2.

# ICV\_EOIR0\_EL1, Interrupt Controller Virtual End Of Interrupt Register 0

The ICV\_EOIR0\_EL1 characteristics are:

## Purpose

A PE writes to this register to inform the CPU interface that it has completed the processing of the specified virtual Group 0 interrupt.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_EOIR0\_EL1 performs the same function as AArch32 System register [ICV\\_EOIR0](#).

## Attributes

ICV\_EOIR0\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_EOIR0\_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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID from the corresponding [ICV\\_IAR0\\_EL1](#) access.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICV\\_CTLR\\_EL1](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

If the [ICV\\_CTLR](#).EOImode bit is 0, a write to this register drops the priority for the virtual interrupt, and also deactivates the virtual interrupt.

If the [ICV\\_CTLR](#).EOImode bit is 1, a write to this register only drops the priority for the virtual interrupt. Software must write to [ICV\\_DIR\\_EL1](#) to deactivate the virtual interrupt.

## Accessing the ICV\_EOIR0\_EL1

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>	op1	CRn	CRm	op2
ICC_EOIR0_EL1	000	1100	1000	001

When HCR\_EL2.FMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_EOIR0\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_EOIR0_EL1</a>	n/a	<a href="#">ICC_EOIR0_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_EOIR0_EL1</a>	<a href="#">ICC_EOIR0_EL1</a>
0	x	0	1	-	<a href="#">ICC_EOIR0_EL1</a>	<a href="#">ICC_EOIR0_EL1</a>	<a href="#">ICC_EOIR0_EL1</a>
1	x	0	1	-	WO	<a href="#">ICC_EOIR0_EL1</a>	<a href="#">ICC_EOIR0_EL1</a>

This table applies to all instructions that can access this register.

ICV\_EOIR0\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.FMO is set to 1.

### Note

When HCR\_EL2.FMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_EOIR0\_EL1 results in an access to [ICC\\_EOIR0\\_EL1](#).

A write to this register must correspond to the most recent valid read by this vPE from a Virtual Interrupt Acknowledge Register, and must correspond to the INTID that was read from [ICV\\_IAR0\\_EL1](#), otherwise the system behavior is UNPREDICTABLE. A valid read is a read that returns a valid INTID that is not a special INTID.

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure write accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

# ICV\_EOIR1\_EL1, Interrupt Controller Virtual End Of Interrupt Register 1

The ICV\_EOIR1\_EL1 characteristics are:

## Purpose

A PE writes to this register to inform the CPU interface that it has completed the processing of the specified virtual Group 1 interrupt.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_EOIR1\_EL1 performs the same function as AArch32 System register [ICV\\_EOIR1](#).

## Attributes

ICV\_EOIR1\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_EOIR1\_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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID from the corresponding [ICV\\_IARI\\_EL1](#) access.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICV\\_CTLR\\_EL1](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

If the [ICV\\_CTLR](#).EOImode bit is 0, a write to this register drops the priority for the virtual interrupt, and also deactivates the virtual interrupt.

If the [ICV\\_CTLR](#).EOImode bit is 1, a write to this register only drops the priority for the virtual interrupt. Software must write to [ICV\\_DIR\\_EL1](#) to deactivate the virtual interrupt.

## Accessing the ICV\_EOIR1\_EL1

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>	op1	CRn	CRm	op2
ICC_EOIR1_EL1	000	1100	1100	001

When HCR\_EL2.IMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_EOIR1\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_EOIR1_EL1</a>	n/a	<a href="#">ICC_EOIR1_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_EOIR1_EL1</a>	<a href="#">ICC_EOIR1_EL1</a>
x	0	0	1	-	<a href="#">ICC_EOIR1_EL1</a>	<a href="#">ICC_EOIR1_EL1</a>	<a href="#">ICC_EOIR1_EL1</a>
x	1	0	1	-	WO	<a href="#">ICC_EOIR1_EL1</a>	<a href="#">ICC_EOIR1_EL1</a>

This table applies to all instructions that can access this register.

ICV\_EOIR1\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.IMO is set to 1.

### Note

When HCR\_EL2.IMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_EOIR1\_EL1 results in an access to [ICC\\_EOIR1\\_EL1](#).

A write to this register must correspond to the most recent valid read by this vPE from a Virtual Interrupt Acknowledge Register, and must correspond to the INTID that was read from [ICV\\_IAR1\\_EL1](#), otherwise the system behavior is UNPREDICTABLE. A valid read is a read that returns a valid INTID that is not a special INTID.

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure write accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

# ICV\_HPPIR0\_EL1, Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0

The ICV\_HPPIR0\_EL1 characteristics are:

## Purpose

Indicates the highest priority pending virtual Group 0 interrupt on the virtual CPU interface.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_HPPIR0\_EL1 performs the same function as AArch32 System register [ICV\\_HPPIR0](#).

## Attributes

ICV\_HPPIR0\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_HPPIR0\_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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the highest priority pending virtual interrupt.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. This special INTID can take the value 1023 only. See Special INTIDs, for more information.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICV\\_CTLR\\_EL1](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

## Accessing the ICV\_HPPIR0\_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>	op1	CRn	CRm	op2
ICC_HPPIR0_EL1	000	1100	1000	010

When HCR\_EL2.FMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_HPPIR0\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_HPPIR0_EL1</a>	n/a	<a href="#">ICC_HPPIR0_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_HPPIR0_EL1</a>	<a href="#">ICC_HPPIR0_EL1</a>
0	x	0	1	-	<a href="#">ICC_HPPIR0_EL1</a>	<a href="#">ICC_HPPIR0_EL1</a>	<a href="#">ICC_HPPIR0_EL1</a>
1	x	0	1	-	RO	<a href="#">ICC_HPPIR0_EL1</a>	<a href="#">ICC_HPPIR0_EL1</a>

This table applies to all instructions that can access this register.

ICV\_HPPIR0\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.FMO is set to 1.

---

### Note

When HCR\_EL2.FMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_HPPIR0\_EL1 results in an access to [ICC\\_HPPIR0\\_EL1](#).

---

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure read accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

# ICV\_HPPIR1\_EL1, Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1

The ICV\_HPPIR1\_EL1 characteristics are:

## Purpose

Indicates the highest priority pending virtual Group 1 interrupt on the virtual CPU interface.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_HPPIR1\_EL1 performs the same function as AArch32 System register [ICV\\_HPPIR1](#).

## Attributes

ICV\_HPPIR1\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_HPPIR1\_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	INTID																											

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the highest priority pending virtual interrupt.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. This special INTID can take the value 1023 only. See Special INTIDs, for more information.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICV\\_CTLR\\_EL1](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

## Accessing the ICV\_HPPIR1\_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>	op1	CRn	CRm	op2
ICC_HPPIR1_EL1	000	1100	1100	010

When HCR\_EL2.IMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_HPPIR1\\_EL1](#).



## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_HPPIR1_EL1</a>	n/a	<a href="#">ICC_HPPIR1_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_HPPIR1_EL1</a>	<a href="#">ICC_HPPIR1_EL1</a>
x	0	0	1	-	<a href="#">ICC_HPPIR1_EL1</a>	<a href="#">ICC_HPPIR1_EL1</a>	<a href="#">ICC_HPPIR1_EL1</a>
x	1	0	1	-	RO	<a href="#">ICC_HPPIR1_EL1</a>	<a href="#">ICC_HPPIR1_EL1</a>

This table applies to all instructions that can access this register.

ICV\_HPPIR1\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.IMO is set to 1.

---

### Note

When HCR\_EL2.IMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_HPPIR1\_EL1 results in an access to [ICC\\_HPPIR1\\_EL1](#).

---

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure read accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

# ICV\_IAR0\_EL1, Interrupt Controller Virtual Interrupt Acknowledge Register 0

The ICV\_IAR0\_EL1 characteristics are:

## Purpose

The PE reads this register to obtain the INTID of the signaled virtual Group 0 interrupt. This read acts as an acknowledge for the interrupt.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_IAR0\_EL1 performs the same function as AArch32 System register [ICV\\_IAR0](#).

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when  $PSTATE.\{I,F\} == \{0,0\}$ ). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. See Observability of the effects of accesses to the GIC registers, for more information.

## Attributes

ICV\_IAR0\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_IAR0\_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													INTID											

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the signaled virtual interrupt.

This is the INTID of the highest priority pending virtual interrupt, if that interrupt is of sufficient priority for it to be signaled to the PE, and if it can be acknowledged.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. This special INTID can take the value 1023 only. See Special INTIDs, for more information.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICV\\_CTLR\\_EL1](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

## Accessing the ICV\_IAR0\_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>	op1	CRn	CRm	op2
ICC_IAR0_EL1	000	1100	1000	000

When HCR\_EL2.FMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_IAR0\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_IAR0_EL1</a>	n/a	<a href="#">ICC_IAR0_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_IAR0_EL1</a>	<a href="#">ICC_IAR0_EL1</a>
0	x	0	1	-	<a href="#">ICC_IAR0_EL1</a>	<a href="#">ICC_IAR0_EL1</a>	<a href="#">ICC_IAR0_EL1</a>
1	x	0	1	-	RO	<a href="#">ICC_IAR0_EL1</a>	<a href="#">ICC_IAR0_EL1</a>

This table applies to all instructions that can access this register.

ICV\_IAR0\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.FMO is set to 1.

### Note

When HCR\_EL2.FMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_IAR0\_EL1 results in an access to [ICC\\_IAR0\\_EL1](#).

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure read accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

# ICV\_IAR1\_EL1, Interrupt Controller Virtual Interrupt Acknowledge Register 1

The ICV\_IAR1\_EL1 characteristics are:

## Purpose

The PE reads this register to obtain the INTID of the signaled virtual Group 1 interrupt. This read acts as an acknowledge for the interrupt.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_IAR1\_EL1 performs the same function as AArch32 System register [ICV\\_IAR1](#).

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that reads of this register are self-synchronising when interrupts are masked by the PE (that is when  $PSTATE.\{I,F\} == \{0,0\}$ ). This ensures that the effect of activating an interrupt on the signaling of interrupt exceptions is observed when a read of this register is architecturally executed so that no spurious interrupt exception occurs if interrupts are unmasked by an instruction immediately following the read. See Observability of the effects of accesses to the GIC registers, for more information.

## Attributes

ICV\_IAR1\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_IAR1\_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													INTID											

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the signaled virtual interrupt.

This is the INTID of the highest priority pending virtual interrupt, if that interrupt is of sufficient priority for it to be signaled to the PE, and if it can be acknowledged.

If the highest priority pending interrupt is not observable, this field contains a special INTID to indicate the reason. This special INTID can take the value 1023 only. See Special INTIDs, for more information.

This field has either 16 or 24 bits implemented. The number of implemented bits can be found in [ICV\\_CTLR\\_EL1](#).IDbits. If only 16 bits are implemented, bits [23:16] of this register are RES0.

## Accessing the ICV\_IAR1\_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>	op1	CRn	CRm	op2
ICC_IAR1_EL1	000	1100	1100	000

When HCR\_EL2.IMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_IAR1\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_IAR1_EL1</a>	n/a	<a href="#">ICC_IAR1_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_IAR1_EL1</a>	<a href="#">ICC_IAR1_EL1</a>
x	0	0	1	-	<a href="#">ICC_IAR1_EL1</a>	<a href="#">ICC_IAR1_EL1</a>	<a href="#">ICC_IAR1_EL1</a>
x	1	0	1	-	RO	<a href="#">ICC_IAR1_EL1</a>	<a href="#">ICC_IAR1_EL1</a>

This table applies to all instructions that can access this register.

ICV\_IAR1\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.IMO is set to 1.

---

### Note

When HCR\_EL2.IMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_IAR1\_EL1 results in an access to [ICC\\_IAR1\\_EL1](#).

---

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure read accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

# ICV\_IGRPEN0\_EL1, Interrupt Controller Virtual Interrupt Group 0 Enable register

The ICV\_IGRPEN0\_EL1 characteristics are:

## Purpose

Controls whether virtual Group 0 interrupts are enabled or not.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_IGRPEN0\_EL1 is architecturally mapped to AArch32 System register [ICV\\_IGRPEN0](#).

## Attributes

ICV\_IGRPEN0\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_IGRPEN0\_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	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\_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>	op1	CRn	CRm	op2
-------------	-----	-----	-----	-----

ICC_IGRPEN0_EL1	000	1100	1100	110
-----------------	-----	------	------	-----

When HCR\_EL2.FMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_IGRPEN0\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_IGRPEN0_EL1</a>	n/a	<a href="#">ICC_IGRPEN0_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_IGRPEN0_EL1</a>	<a href="#">ICC_IGRPEN0_EL1</a>
0	x	0	1	-	<a href="#">ICC_IGRPEN0_EL1</a>	<a href="#">ICC_IGRPEN0_EL1</a>	<a href="#">ICC_IGRPEN0_EL1</a>
1	x	0	1	-	RW	<a href="#">ICC_IGRPEN0_EL1</a>	<a href="#">ICC_IGRPEN0_EL1</a>

This table applies to all instructions that can access this register.

ICV\_IGRPEN0\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.FMO is set to 1.

### Note

When HCR\_EL2.FMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_IGRPEN0\_EL1 results in an access to [ICC\\_IGRPEN0\\_EL1](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL0==1, Non-secure accesses to this register from EL1 are trapped to EL2.

# ICV\_IGRPEN1\_EL1, Interrupt Controller Virtual Interrupt Group 1 Enable register

The ICV\_IGRPEN1\_EL1 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

AArch64 System register ICV\_IGRPEN1\_EL1 is architecturally mapped to AArch32 System register [ICV\\_IGRPEN1](#).

## Attributes

ICV\_IGRPEN1\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_IGRPEN1\_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	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\_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>	op1	CRn	CRm	op2
-------------	-----	-----	-----	-----



ICC_IGRPEN1_EL1	000	1100	1100	111
-----------------	-----	------	------	-----

When HCR\_EL2.IMO is set to 0, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_IGRPEN1\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_IGRPEN1_EL1</a>	n/a	<a href="#">ICC_IGRPEN1_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_IGRPEN1_EL1</a>	<a href="#">ICC_IGRPEN1_EL1</a>
x	0	0	1	-	<a href="#">ICC_IGRPEN1_EL1</a>	<a href="#">ICC_IGRPEN1_EL1</a>	<a href="#">ICC_IGRPEN1_EL1</a>
x	1	0	1	-	RW	<a href="#">ICC_IGRPEN1_EL1</a>	<a href="#">ICC_IGRPEN1_EL1</a>

This table applies to all instructions that can access this register.

ICV\_IGRPEN1\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.IMO is set to 1.

### Note

When HCR\_EL2.IMO is set to 0, at Non-secure EL1, the instruction encoding used to access ICV\_IGRPEN1\_EL1 results in an access to [ICC\\_IGRPEN1\\_EL1](#).

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TALL1==1, Non-secure accesses to this register from EL1 are trapped to EL2.

# ICV\_PMR\_EL1, Interrupt Controller Virtual Interrupt Priority Mask Register

The ICV\_PMR\_EL1 characteristics are:

## Purpose

Provides a virtual interrupt priority filter. Only virtual interrupts with a higher priority than the value in this register are signaled to the PE.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_PMR\_EL1 is architecturally mapped to AArch32 System register [ICV\\_PMR](#).

To allow software to ensure appropriate observability of actions initiated by GIC register accesses, the PE and CPU interface logic must ensure that writes to this register are self-synchronising. This ensures that no interrupts below the written PMR value will be taken after a write to this register is architecturally executed. See Observability of the effects of accesses to the GIC registers, for more information.

## Attributes

ICV\_PMR\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_PMR\_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	Priority									

### Bits [31:8]

Reserved, RES0.

### Priority, bits [7:0]

The priority mask level for the virtual CPU interface. If the priority of a virtual interrupt is higher than the value indicated by this field, the interface signals the virtual interrupt to the PE.

The possible priority field values are as follows:

Implemented priority bits	Possible priority field values	Number of priority levels
[7:0]	0x00-0xFF (0-255), all values	256
[7:1]	0x00-0xFE (0-254), even values only	128
[7:2]	0x00-0xFC (0-252), in steps of 4	64
[7:3]	0x00-0xF8 (0-248), in steps of 8	32
[7:4]	0x00-0xF0 (0-240), in steps of 16	16

Unimplemented priority bits are RAZ/WI.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the ICV\_PMR\_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>	op1	CRn	CRm	op2
ICC_PMR_EL1	000	0100	0110	000

When HCR\_EL2.{FMO, IMO} == {0, 0}, execution of this encoding at Non-secure EL1 results in an access to [ICC\\_PMR\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_PMR_EL1</a>	n/a	<a href="#">ICC_PMR_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_PMR_EL1</a>	<a href="#">ICC_PMR_EL1</a>
x	1	0	1	-	RW	<a href="#">ICC_PMR_EL1</a>	<a href="#">ICC_PMR_EL1</a>
1	x	0	1	-	RW	<a href="#">ICC_PMR_EL1</a>	<a href="#">ICC_PMR_EL1</a>
0	0	0	1	-	<a href="#">ICC_PMR_EL1</a>	<a href="#">ICC_PMR_EL1</a>	<a href="#">ICC_PMR_EL1</a>

This table applies to all instructions that can access this register.

ICV\_PMR\_EL1 is only accessible at Non-secure EL1 when HCR\_EL2.{FMO, IMO} != {0, 0}.

### Note

When HCR\_EL2.{FMO, IMO} == {0, 0}, at Non-secure EL1, the instruction encoding used to access ICV\_PMR\_EL1 results in an access to [ICC\\_PMR\\_EL1](#).

## 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure accesses to this register from EL1 are trapped to EL1.

When SCR\_EL3.NS==1 :

- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure accesses to this register from EL1 are trapped to EL2.

# ICV\_RPR\_EL1, Interrupt Controller Virtual Running Priority Register

The ICV\_RPR\_EL1 characteristics are:

## Purpose

Indicates the Running priority of the virtual CPU interface.

This register is part of:

- The GIC system registers functional group.
- The GIC virtual interface control registers functional group.

## Configuration

AArch64 System register ICV\_RPR\_EL1 performs the same function as AArch32 System register [ICV\\_RPR](#).

## Attributes

ICV\_RPR\_EL1 is a 32-bit register.

## Field descriptions

The ICV\_RPR\_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	Priority							

### Bits [31:8]

Reserved, RES0.

### Priority, bits [7:0]

The current running priority on the virtual CPU interface. This is the group priority of the current active virtual interrupt.

If there are no active interrupts on the virtual CPU interface, or all active interrupts have undergone a priority drop, the value returned is the Idle priority.

The priority returned is the group priority as if the BPR for the current Exception level and Security state was set to the minimum value of BPR for the number of implemented priority bits.

---

#### Note

If 8 bits of priority are implemented the group priority is bits[7:1] of the priority.

---

## Accessing the ICV\_RPR\_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>	op1	CRn	CRm	op2
-------------	-----	-----	-----	-----

ICC_RPR_EL1	000	1100	1011	011
-------------	-----	------	------	-----

When  $\text{HCR\_EL2}\{FMO, IMO\} == \{0, 0\}$ , execution of this encoding at Non-secure EL1 results in an access to [ICC\\_RPR\\_EL1](#).

## Accessibility

The register is accessible as follows:

Control				Accessibility			
FMO	IMO	TGE	NS	EL0	EL1	EL2	EL3
x	x	x	0	-	<a href="#">ICC_RPR_EL1</a>	n/a	<a href="#">ICC_RPR_EL1</a>
x	x	1	1	-	n/a	<a href="#">ICC_RPR_EL1</a>	<a href="#">ICC_RPR_EL1</a>
x	1	0	1	-	RO	<a href="#">ICC_RPR_EL1</a>	<a href="#">ICC_RPR_EL1</a>
1	x	0	1	-	RO	<a href="#">ICC_RPR_EL1</a>	<a href="#">ICC_RPR_EL1</a>
0	0	0	1	-	<a href="#">ICC_RPR_EL1</a>	<a href="#">ICC_RPR_EL1</a>	<a href="#">ICC_RPR_EL1</a>

This table applies to all instructions that can access this register.

ICV\_RPR\_EL1 is only accessible at Non-secure EL1 when  $\text{HCR\_EL2}\{FMO, IMO\} \neq \{0, 0\}$ .

---

### Note

When  $\text{HCR\_EL2}\{FMO, IMO\} == \{0, 0\}$ , at Non-secure EL1, the instruction encoding used to access ICV\_RPR\_EL1 results in an access to [ICC\\_RPR\\_EL1](#).

---

If there are no active interrupts on the virtual CPU interface, or all active interrupts have undergone a priority drop, the value returned is the Idle priority.

Software cannot determine the number of implemented priority bits from a read of this register.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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 [ICC\\_SRE\\_EL1](#).SRE==0, Non-secure read accesses to this register from EL1 are trapped to EL1.

When  $\text{SCR\_EL3.NS}==1$  :

- If [ICH\\_HCR\\_EL2](#).TC==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

# ID\_AA64AFR0\_EL1, AArch64 Auxiliary Feature Register 0

The ID\_AA64AFR0\_EL1 characteristics are:

## Purpose

Provides information about the IMPLEMENTATION DEFINED features of the PE 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\_AA64AFR0\_EL1 is a 64-bit register.

## Field descriptions

The ID\_AA64AFR0\_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
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
IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6

### Bits [63:32]

Reserved, RES0.

### IMPLEMENTATION DEFINED, bits [31:28]

IMPLEMENTATION DEFINED.

### IMPLEMENTATION DEFINED, bits [27:24]

IMPLEMENTATION DEFINED.

### IMPLEMENTATION DEFINED, bits [23:20]

IMPLEMENTATION DEFINED.

### IMPLEMENTATION DEFINED, bits [19:16]

IMPLEMENTATION DEFINED.

### IMPLEMENTATION DEFINED, bits [15:12]

IMPLEMENTATION DEFINED.

**IMPLEMENTATION DEFINED, bits [11:8]**

IMPLEMENTATION DEFINED.

**IMPLEMENTATION DEFINED, bits [7:4]**

IMPLEMENTATION DEFINED.

**IMPLEMENTATION DEFINED, bits [3:0]**

IMPLEMENTATION DEFINED.

**Accessing the ID\_AA64AFR0\_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_AA64AFR0_EL1	11	000	0000	0101	100

**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<sup>®</sup> 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.

# ID\_AA64AFR1\_EL1, AArch64 Auxiliary Feature Register 1

The ID\_AA64AFR1\_EL1 characteristics are:

## Purpose

Reserved for future expansion of information about the IMPLEMENTATION DEFINED features of the PE 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\_AA64AFR1\_EL1 is a 64-bit register.

## Field descriptions

The ID\_AA64AFR1\_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
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 [63:0]

Reserved, RES0.

## Accessing the ID\_AA64AFR1\_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_AA64AFR1_EL1	11	000	0000	0101	101

## 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- If [HCR\\_EL2.TID3](#)=1, Non-secure read accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ID\_AA64DFR0\_EL1, AArch64 Debug Feature Register 0

The ID\_AA64DFR0\_EL1 characteristics are:

## Purpose

Provides top level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

## Configuration

The external register [EDDFR](#) gives information from this register.

## Attributes

ID\_AA64DFR0\_EL1 is a 64-bit register.

## Field descriptions

The ID\_AA64DFR0\_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
CTX_CMPs				0	0	0	0	WRPs				0	0	0	0	BRPs				PMUVer				TraceVer				DebugVer			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:36]

Reserved, RES0.

### PMSVer, bits [35:32]

In ARMv8.2:

Statistical Profiling Extension version. When the Statistical Profiling Extension is implemented, the defined values of this field are:

PMSVer	Meaning
0000	No Statistical Profiling extension.
0001	Version 1 of the Statistical Profiling extension present.

All other values are reserved.

When the Statistical Profiling Extension is not implemented this field is reserved, RES0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

### CTX\_CMPs, bits [31:28]

Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints.

**Bits [27:24]**

Reserved, RES0.

**WRPs, bits [23:20]**

Number of watchpoints, minus 1. The value of 0b0000 is reserved.

**Bits [19:16]**

Reserved, RES0.

**BRPs, bits [15:12]**

Number of breakpoints, minus 1. The value of 0b0000 is reserved.

**PMUVer, bits [11:8]**

Performance Monitors Extension version. Indicates whether System register interface to Performance Monitors extension is implemented. Defined values are:

PMUVer	Meaning
0000	Performance Monitors Extension System registers not implemented.
0001	Performance Monitors Extension System registers implemented, PMUv3.
0100	Performance Monitors Extension System registers implemented, PMUv3, with a 16-bit evtCount field, and if EL2 is implemented, the addition of the MDCR_EL2.HPMD bit.
1111	IMPLEMENTATION DEFINED form of performance monitors supported, PMUv3 not supported.

All other values are reserved.

In ARMv8.0 the permitted values are 0b0000, 0b0001 and 0b1111.

From ARMv8.1 the permitted values are 0b0000, 0b0100 and 0b1111.

**TraceVer, bits [7:4]**

Trace support. Indicates whether System register interface to a trace macrocell is implemented. Defined values are:

TraceVer	Meaning
0000	Trace macrocell System registers not implemented.
0001	Trace macrocell System registers implemented.

All other values are reserved.

A value of 0b0000 only indicates that no System register interface to a trace macrocell is implemented. A trace macrocell might nevertheless be implemented without a System register interface.

**DebugVer, bits [3:0]**

Debug architecture version. Indicates presence of ARMv8 debug architecture.

DebugVer	Meaning
0110	ARMv8 debug architecture.
0111	ARMv8 debug architecture with Virtualization Host Extensions.
1000	ARMv8.2 debug architecture

All other values are reserved.

In an ARMv8.0 implementation, the only permitted value is 0b0110.

In an ARMv8.1 implementation that includes ARMv8.1-VHE, the only permitted value is 0b0111.

In an ARMv8.1 implementation that does not include ARMv8.1-VHE, the permitted values are 0b0110 and 0b0111.

In an ARMv8.2 implementation, the only permitted value is 0b1000.

## Accessing the ID\_AA64DFR0\_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_AA64DFR0_EL1	11	000	0000	0101	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.

# ID\_AA64DFR1\_EL1, AArch64 Debug Feature Register 1

The ID\_AA64DFR1\_EL1 characteristics are:

## Purpose

Reserved for future expansion of top level information about the debug system in AArch64 state.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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\_AA64DFR1\_EL1 is a 64-bit register.

## Field descriptions

The ID\_AA64DFR1\_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
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 [63:0]

Reserved, RES0.

## Accessing the ID\_AA64DFR1\_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_AA64DFR1_EL1	11	000	0000	0101	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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- If [HCR\\_EL2.TID3](#)=1, Non-secure read accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ID\_AA64ISAR0\_EL1, AArch64 Instruction Set Attribute Register 0

The ID\_AA64ISAR0\_EL1 characteristics are:

## Purpose

Provides information about the instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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\_AA64ISAR0\_EL1 is a 64-bit register.

## Field descriptions

The ID\_AA64ISAR0\_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
RDM				0	0	0	0	Atomic				CRC32				SHA2				SHA1				AES				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 [63:32]

Reserved, RES0.

### RDM, bits [31:28]

In ARMv8.2 and ARMv8.1:

SQRDMLAH and SQRDMLSH instructions implemented in AArch64 state. Defined values are:

RDM	Meaning
0000	No SQRDMLAH and SQRDMLSH instructions implemented.
0001	SQRDMLAH and SQRDMLSH instructions implemented.

All other values are reserved.

In ARMv8.0 the only permitted value is 0000.

From ARMv8.1 the only permitted value is 0001. This feature is identified by the name ARMv8.1-RDMA.

### In ARMv8.0:

Reserved, RES0.

**Bits [27:24]**

Reserved, RES0.

**Atomic, bits [23:20]**

In ARMv8.2 and ARMv8.1:

Atomic instructions implemented in AArch64 state. Defined values are:

Atomic	Meaning
0000	No Atomic instructions implemented.
0010	LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions implemented.

All other values are reserved.

In ARMv8.0 the only permitted value is 0000.

From ARMv8.1 the only permitted value is 0010. This feature is identified by the name ARMv8.1-LSE.

In ARMv8.0:

Reserved, RES0.

**CRC32, bits [19:16]**

CRC32 instructions implemented in AArch64 state. Defined values are:

CRC32	Meaning
0000	No CRC32 instructions implemented.
0001	CRC32B, CRC32H, CRC32W, CRC32X, CRC32CB, CRC32CH, CRC32CW, and CRC32CX instructions implemented.

All other values are reserved.

In ARMv8.0 the permitted values are 0000 and 0001.

From ARMv8.1 the only permitted value is 0001.

**SHA2, bits [15:12]**

SHA2 instructions implemented in AArch64 state. Defined values are:

SHA2	Meaning
0000	No SHA2 instructions implemented.
0001	SHA256H, SHA256H2, SHA256SU0, and SHA256SU1 instructions implemented.

All other values are reserved.

**SHA1, bits [11:8]**

SHA1 instructions implemented in AArch64 state. Defined values are:

SHA1	Meaning
0000	No SHA1 instructions implemented.
0001	SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 instructions implemented.

All other values are reserved.



**AES, bits [7:4]**

AES instructions implemented in AArch64 state. Defined values are:

AES	Meaning
0000	No AES instructions implemented.
0001	AESE, AESD, AESMC, and AESIMC instructions implemented.
0010	As for 0001, plus PMULL/PMULL2 instructions operating on 64-bit data quantities.

All other values are reserved.

**Bits [3:0]**

Reserved, RES0.

**Accessing the ID\_AA64ISAR0\_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_AA64ISAR0_EL1	11	000	0000	0110	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.

# ID\_AA64ISAR1\_EL1, AArch64 Instruction Set Attribute Register 1

The ID\_AA64ISAR1\_EL1 characteristics are:

## Purpose

Reserved for future expansion of the information about the instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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
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 [63:4]

Reserved, RES0.

### DPB, bits [3:0]

In ARMv8.2:

Indicates support for the [DC CVAP](#) instruction in AArch64 state. Defined values are:

DPB	Meaning
0000	<a href="#">DC CVAP</a> not supported.
0001	<a href="#">DC CVAP</a> supported.

All other values are reserved.

In ARMv8.0 and ARMv8.1 the only permitted value is 0000.

From ARMv8.2 the only permitted value is 0001. This feature is identified by the name ARMv8.2-DCPoP.

### In ARMv8.1 and ARMv8.0:

Reserved, 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.

# ID\_AA64MMFR0\_EL1, AArch64 Memory Model Feature Register 0

The ID\_AA64MMFR0\_EL1 characteristics are:

## Purpose

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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\_AA64MMFR0\_EL1 is a 64-bit register.

## Field descriptions

The ID\_AA64MMFR0\_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
TGran4				TGran64				TGran16				BigEndEL0				SNSMem				BigEnd				ASIDBits				PARange			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:32]

Reserved, RES0.

### TGran4, bits [31:28]

Support for 4KB memory translation granule size. Defined values are:

TGran4	Meaning
0000	4KB granule supported.
1111	4KB granule not supported.

All other values are reserved.

### TGran64, bits [27:24]

Support for 64KB memory translation granule size. Defined values are:

TGran64	Meaning
0000	64KB granule supported.
1111	64KB granule not supported.

All other values are reserved.

**TGran16, bits [23:20]**

Support for 16KB memory translation granule size. Defined values are:

<b>TGran16</b>	<b>Meaning</b>
0000	16KB granule not supported.
0001	16KB granule supported.

All other values are reserved.

**BigEndEL0, bits [19:16]**

Mixed-endian support at EL0 only. Defined values are:

<b>BigEndEL0</b>	<b>Meaning</b>
0000	No mixed-endian support at EL0. The <a href="#">SCTLR_EL1.E0E</a> bit has a fixed value.
0001	Mixed-endian support at EL0. The <a href="#">SCTLR_EL1.E0E</a> bit can be configured.

All other values are reserved.

This field is invalid and is RES0 if the BigEnd field, bits [11:8], is not 0000.

**SNSMem, bits [15:12]**

Secure versus Non-secure Memory distinction. Defined values are:

<b>SNSMem</b>	<b>Meaning</b>
0000	Does not support a distinction between Secure and Non-secure Memory.
0001	Does support a distinction between Secure and Non-secure Memory.

All other values are reserved.

**BigEnd, bits [11:8]**

Mixed-endian configuration support. Defined values are:

<b>BigEnd</b>	<b>Meaning</b>
0000	No mixed-endian support. The SCTLR_ELx.EE bits have a fixed value. See the BigEndEL0 field, bits[19:16], for whether EL0 supports mixed-endian.
0001	Mixed-endian support. The SCTLR_ELx.EE and <a href="#">SCTLR_EL1.E0E</a> bits can be configured.

All other values are reserved.

**ASIDBits, bits [7:4]**

Number of ASID bits. Defined values are:

<b>ASIDBits</b>	<b>Meaning</b>
0000	8 bits.
0010	16 bits.

All other values are reserved.

**PARange, bits [3:0]**

Physical Address range supported. Defined values are:

PARange	Meaning
0000	32 bits, 4GB.
0001	36 bits, 64GB.
0010	40 bits, 1TB.
0011	42 bits, 4TB.
0100	44 bits, 16TB.
0101	48 bits, 256TB.
0110	52 bits, 4PB.

All other values are reserved.

In all ARMv8 implementations the values 0000, 0001, 0010, 0011, 0100 and 0101 are permitted.

From ARMv8.1 the value 0110 is permitted and indicates that ARMv8.2-LPA is implemented.

## Accessing the ID\_AA64MMFR0\_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_AA64MMFR0_EL1	11	000	0000	0111	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<sup>®</sup> 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.

# ID\_AA64MMFR1\_EL1, AArch64 Memory Model Feature Register 1

The ID\_AA64MMFR1\_EL1 characteristics are:

## Purpose

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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\_AA64MMFR1\_EL1 is a 64-bit register.

## Field descriptions

The ID\_AA64MMFR1\_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
XNX				SpecSEI				PAN				LO				HPDS				VH				VMIDBits				HAFDBS			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:32]

Reserved, RES0.

### XNX, bits [31:28]

In ARMv8.2:

Indicates support for Execute Never control distinction at stage 2 bit. Defined values are:

XNX	Meaning
0000	Distinction between EL0 and EL1 execute permission at stage 2 not supported.
0001	Distinction between EL0 and EL1 execute permission at stage 2 supported.

All other values are reserved.

In ARMv8.0 and ARMv8.1, the only permitted value is 0000.

From ARMv8.2, the only permitted value is 0001. This feature is identified by the name ARMv8.2-TTS2UXN.

### In ARMv8.1 and ARMv8.0:

Reserved, RES0.

**SpecSEI, bits [27:24]**

Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches. The defined values of this field are:

SpecSEI	Meaning
0000	The PE never generates an SError interrupt due to an external abort on a speculative read.
0001	The PE might generate an SError interrupt due to an external abort on a speculative read.

All other values are reserved.

When the RAS Extension is not implemented, this field is RAZ.

**PAN, bits [23:20]**

In ARMv8.2 and ARMv8.1:

Privileged Access Never. Indicates support for the PAN bit in PSTATE, [SPSR\\_EL1](#), [SPSR\\_EL2](#), [SPSR\\_EL3](#), and [DSPSR\\_EL0](#). Defined values are:

PAN	Meaning
0000	PAN not supported.
0001	PAN supported.
0010	PAN supported and <a href="#">AT S1E1RP</a> and <a href="#">AT S1E1WP</a> instructions supported.

All other values are reserved.

In ARMv8.0 the only permitted value is 0000.

In ARMv8.1 the only permitted value is 0001. This feature is identified by the name ARMv8.1-PAN.

From ARMv8.2, the only permitted value is 0010. This feature is identified by the name ARMv8.2-ATS1E1.

In ARMv8.0:

Reserved, RES0.

**LO, bits [19:16]**

In ARMv8.2 and ARMv8.1:

LORegions. Indicates support for LORegions. Defined values are:

LO	Meaning
0000	LORegions not supported.
0001	LORegions supported.

All other values are reserved.

In ARMv8.0 the only permitted value is 0000.

From ARMv8.1, the only permitted value is 0001. This feature is identified by the name ARMv8.1-LOR.

In ARMv8.0:

Reserved, RES0.

**HPDS, bits [15:12]**

In ARMv8.2 and ARMv8.1:

Hierarchical permission disables bits in translation tables. Defined values are:



HPDS	Meaning
0000	Disabling of hierarchical controls not supported.
0001	Disabling of hierarchical controls supported using <a href="#">TCR_EL1</a> .HPD0, <a href="#">TCR_EL1</a> .HPD1, <a href="#">TCR_EL2</a> .HPD, and <a href="#">TCR_EL3</a> .HPD bits.
0010	Disabling of hierarchical controls supported using the <a href="#">TCR_EL1</a> .HPD0, <a href="#">TCR_EL1</a> .HPD1, <a href="#">TCR_EL2</a> .HPD, and <a href="#">TCR_EL3</a> .HPD bits, and hardware allocation of bits[62:59] of the last level page table descriptor for IMPLEMENTATION DEFINED use.

All other values are reserved.

In ARMv8.0 the only permitted value is 0000.

In ARMv8.1 the only permitted value is 0001. This feature is identified by the name ARMv8.1-HPD.

From ARMv8.2, the permitted values are 0001 and 0010. This feature is identified by the name ARMv8.2-TTPBHA.

#### In ARMv8.0:

Reserved, RES0.

#### VH, bits [11:8]

##### In ARMv8.2 and ARMv8.1:

Virtualization Host Extensions. Defined values are:

VH	Meaning
0000	Virtualization Host Extensions not supported.
0001	Virtualization Host Extensions supported.

All other values are reserved.

In ARMv8.0 the only permitted value is 0000.

From ARMv8.1, the only permitted value is 0001. This feature is identified by the name ARMv8.1-VHE.

#### In ARMv8.0:

Reserved, RES0.

#### VMIDBits, bits [7:4]

##### In ARMv8.2 and ARMv8.1:

Number of VMID bits. Defined values are:

VMIDBits	Meaning
0000	8 bits
0010	16 bits

All other values are reserved.

In ARMv8.0 the only permitted value is 0000.

From ARMv8.1, the permitted values are 0000 and 0010.

#### In ARMv8.0:

Reserved, RES0.

**HAFDBS, bits [3:0]**

In ARMv8.2 and ARMv8.1:

Hardware updates to Access flag and Dirty state in translation tables. Defined values are:

HAFDBS	Meaning
0000	No hardware update of the Access flag and dirty state is supported in hardware.
0001	Hardware update of the Access flag is supported in hardware.
0010	Hardware update of both the Access flag and dirty state is supported in hardware.

All other values are reserved.

From ARMv8.1, the permitted values are 0000, 0001, and 0010. This feature is identified by the name ARMv8.1-VHE.

In ARMv8.0:

Reserved, RES0.

**Accessing the ID\_AA64MMFR1\_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_AA64MMFR1_EL1	11	000	0000	0111	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.

# ID\_AA64MMFR2\_EL1, AArch64 Memory Model Feature Register 2

The ID\_AA64MMFR2\_EL1 characteristics are:

## Purpose

Provides information about the implemented memory model and memory management support in AArch64 state.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

## Configuration

This register is introduced in ARMv8.2.

## Attributes

ID\_AA64MMFR2\_EL1 is a 64-bit register.

## Field descriptions

The ID\_AA64MMFR2\_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
0	0	0	0	0	0	0	0	0	0	0	0	VARange				IESB				LSM				UAO				CnP			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:20]

Reserved, RES0.

### VARange, bits [19:16]

Indicates support for a larger virtual address. Defined values are:

VARange	Meaning
0000	48 bits of VA for each translation table page register are supported.
0001	52 bits of VA for each translation table page register are supported.

All other values are reserved.

From ARMv8.2, the permitted values are 0000 and 0001. This feature is identified by the name ARMv8.2-LVA.

### IESB, bits [15:12]

Indicates whether the implicit Error Synchronization Barrier operations are implemented. Defined values are:

IESB	Meaning
0000	SCTLR_ELx.IESB implicit ErrorSynchronizationBarrier control not implemented.
0001	SCTLR_ELx.IESB implicit ErrorSynchronizationBarrier control implemented.

All other values are reserved.

**LSM, bits [11:8]**

Indicates support for LSMAOE and nTLSMD bits in [SCTLR\\_EL1](#) and [SCTLR\\_EL2](#). Defined values are:

LSM	Meaning
0000	LSMAOE and nTLSMD bits not supported.
0001	LSMAOE and nTLSMD bits supported.

All other values are reserved.

From ARMv8.2, the permitted values are 0000 and 0001. This feature is identified by the name ARMv8.2-LSMAOC.

**UAO, bits [7:4]**

User Access Override. Defined values are:

UAO	Meaning
0000	UAO not supported.
0001	UAO supported.

All other values are reserved.

From ARMv8.2, the only permitted value is 0001. This feature is identified by the name ARMv8.2-UAO.

**CnP, bits [3:0]**

Common not Private translations. Defined values are:

CnP	Meaning
0000	Common not Private translations not supported.
0001	Common not Private translations supported.

All other values are reserved.

From ARMv8.2, the only permitted value is 0001. This feature is identified by the name ARMv8.2-TTCNP.

**Accessing the ID\_AA64MMFR2\_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_AA64MMFR2_EL1	11	000	0000	0111	010

**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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- If [HCR\\_EL2.TID3](#)=1, Non-secure read accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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
																												EL0			

### Bits [63:36]

Reserved, RES0.

### SVE, bits [35:32]

In 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 SIMD and SIMD operations: <ul style="list-style-type: none"> <li>Integer byte, halfword, word and doubleword element operations.</li> <li>Single-precision and double-precision floating-point arithmetic.</li> <li>Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.</li> </ul>
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"> <li>Single-precision and double-precision floating-point types.</li> <li>Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.</li> </ul>
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, 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- If [HCR\\_EL2.TID3](#)=1, Non-secure read accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ID\_AA64PFR1\_EL1, AArch64 Processor Feature Register 1

The ID\_AA64PFR1\_EL1 characteristics are:

## Purpose

Reserved for future expansion of information about implemented PE features in AArch64 state.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' 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\_AA64PFR1\_EL1 is a 64-bit register.

## Field descriptions

The ID\_AA64PFR1\_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
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 [63:0]

Reserved, RES0.

## Accessing the ID\_AA64PFR1\_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_AA64PFR1_EL1	11	000	0000	0100	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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- If [HCR\\_EL2.TID3](#)=1, Non-secure read accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ID\_AFR0\_EL1, AArch32 Auxiliary Feature Register 0

The ID\_AFR0\_EL1 characteristics are:

## Purpose

Provides information about the IMPLEMENTATION DEFINED features of the PE in AArch32 state.

Must be interpreted with the Main ID Register, [MIDR\\_EL1](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register ID\_AFR0\_EL1 is architecturally mapped to AArch32 System register [ID\\_AFR0](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_AFR0\_EL1 is a 32-bit register.

## Field descriptions

The ID\_AFR0\_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	IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED				IMPLEMENTATION DEFINED			

### Bits [31:16]

Reserved, RES0.

### IMPLEMENTATION DEFINED, bits [15:12]

IMPLEMENTATION DEFINED.

### IMPLEMENTATION DEFINED, bits [11:8]

IMPLEMENTATION DEFINED.

### IMPLEMENTATION DEFINED, bits [7:4]

IMPLEMENTATION DEFINED.

### IMPLEMENTATION DEFINED, bits [3:0]

IMPLEMENTATION DEFINED.

## Accessing the ID\_AFR0\_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_AFR0_EL1	11	000	0000	0001	011

## 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.

# ID\_DFR0\_EL1, AArch32 Debug Feature Register 0

The ID\_DFR0\_EL1 characteristics are:

## Purpose

Provides top level information about the debug system in AArch32 state.

Must be interpreted with the Main ID Register, [MIDR\\_EL1](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register ID\_DFR0\_EL1 is architecturally mapped to AArch32 System register [ID\\_DFR0](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_DFR0\_EL1 is a 32-bit register.

## Field descriptions

The ID\_DFR0\_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																												

### Bits [31:28]

Reserved, RES0.

### PerfMon, bits [27:24]

Performance Monitors. Support for System registers-based ARM Performance Monitors Extension, using registers in the coproc == 1111 encoding space, for A and R profile processors. Defined values are:

PerfMon	Meaning
0000	Performance Monitors Extension System registers not implemented.
0001	Support for Performance Monitors Extension version 1 (PMUv1) System registers.
0010	Support for Performance Monitors Extension version 2 (PMUv2) System registers.
0011	Support for Performance Monitors Extension version 3 (PMUv3) System registers.
0100	Support for Performance Monitors Extension version 3 (PMUv3) System registers, with a 16-bit evtCount field.
1111	IMPLEMENTATION DEFINED form of Performance Monitors System registers supported. PMUv3 not supported.

All other values are reserved.

In ARMv8.0 the permitted values are 0000, 0011, and 1111.

From ARMv8.1 the permitted values are 0000, 0100, and 1111.

In ARMv7, the value 0000 can mean that PMUv1 is implemented. PMUv1 is not permitted in an ARMv8 implementation.

### MProfDbg, bits [23:20]

M Profile Debug. Support for memory-mapped debug model for M profile processors. Defined values are:

MProfDbg	Meaning
0000	Not supported.
0001	Support for M profile Debug architecture, with memory-mapped access.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### MMapTrc, bits [19:16]

Memory Mapped Trace. Support for memory-mapped trace model. Defined values are:

MMapTrc	Meaning
0000	Not supported.
0001	Support for ARM trace architecture, with memory-mapped access.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

In the Trace registers, the ETMIDR gives more information about the implementation.

### CopTrc, bits [15:12]

Support for System registers-based trace model, using registers in the coproc == 1110 encoding space. Defined values are:

CopTrc	Meaning
0000	Not supported.
0001	Support for ARM trace architecture, with System registers access.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

In the Trace registers, the ETMIDR gives more information about the implementation.

### MMapDbg, bits [11:8]

Memory Mapped Debug. Support for v7 memory-mapped debug model, for A and R profile processors.

In ARMv8-A this field is RES0.

The optional memory map defined by ARMv8 is not compatible with ARMv7.

### CopSDBG, bits [7:4]

Support for a System registers-based Secure debug model, using registers in the coproc = 1110 encoding space, for an A profile processor that includes EL3.

If EL3 is not implemented and the implemented Security state is Non-Secure state, this field is RES0. Otherwise, this field reads the same as bits [3:0].

### CopDbg, bits [3:0]

Support for System registers-based debug model, using registers in the coproc == 1110 encoding space, for A and R profile processors. Defined values are:

CopDbg	Meaning
0000	Not supported.
0010	Support for ARMv6, v6 Debug architecture, with System registers access.
0011	Support for ARMv6, v6.1 Debug architecture, with System registers access.
0100	Support for ARMv7, v7 Debug architecture, with System registers access.
0101	Support for ARMv7, v7.1 Debug architecture, with System registers access.
0110	Support for ARMv8 debug architecture, with System registers access.
0111	Support for ARMv8 debug architecture, with System registers access, and Virtualization Host extensions.
1000	Support for ARMv8.2 debug architecture.

All other values are reserved.

In an ARMv8.0 implementation, the only permitted value is 0b0110.

In an ARMv8.1 implementation that does not include ARMv8.1-VHE, the permitted values are 0b0110 and 0b0111.

In an ARMv8.1 implementation that includes ARMv8.1-VHE, the only permitted value is 0b0111.

In an ARMv8.2 implementation, the only permitted value is 0b1000.

## Accessing the ID\_DFR0\_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_DFR0_EL1	11	000	0000	0001	010

## 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.



# ID\_ISAR0\_EL1, AArch32 Instruction Set Attribute Register 0

The ID\_ISAR0\_EL1 characteristics are:

## Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID\\_ISAR1\\_EL1](#), [ID\\_ISAR2\\_EL1](#), [ID\\_ISAR3\\_EL1](#), [ID\\_ISAR4\\_EL1](#), and [ID\\_ISAR5\\_EL1](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register ID\_ISAR0\_EL1 is architecturally mapped to AArch32 System register [ID\\_ISAR0](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_ISAR0\_EL1 is a 32-bit register.

## Field descriptions

The ID\_ISAR0\_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		Divide			Debug				Coproc				CmpBranch				BitField				BitCount				Swap			

### Bits [31:28]

Reserved, RES0.

### Divide, bits [27:24]

Indicates the implemented Divide instructions. Defined values are:

Divide	Meaning
0000	None implemented.
0001	Adds SDIV and UDIV in the T32 instruction set.
0010	As for 0001, and adds SDIV and UDIV in the A32 instruction set.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

### Debug, bits [23:20]

Indicates the implemented Debug instructions. Defined values are:

Debug	Meaning
0000	None implemented.
0001	Adds BKPT.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Coproc, bits [19:16]

Indicates the implemented System register access instructions. Defined values are:

Coproc	Meaning
0000	None implemented, except for instructions separately attributed by the architecture to provide access to AArch32 System registers and System instructions.
0001	Adds generic CDP, LDC, MCR, MRC, and STC.
0010	As for 0001, and adds generic CDP2, LDC2, MCR2, MRC2, and STC2.
0011	As for 0010, and adds generic MCRR and MRRC.
0100	As for 0011, and adds generic MCRR2 and MRRC2.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### CmpBranch, bits [15:12]

Indicates the implemented combined Compare and Branch instructions in the T32 instruction set. Defined values are:

CmpBranch	Meaning
0000	None implemented.
0001	Adds CBNZ and CBZ.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### BitField, bits [11:8]

Indicates the implemented BitField instructions. Defined values are:

BitField	Meaning
0000	None implemented.
0001	Adds BFC, BFI, SBFX, and UBFX.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### BitCount, bits [7:4]

Indicates the implemented Bit Counting instructions. Defined values are:

BitCount	Meaning
0000	None implemented.
0001	Adds CLZ.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Swap, bits [3:0]

Indicates the implemented Swap instructions in the A32 instruction set. Defined values are:

Swap	Meaning
0000	None implemented.
0001	Adds SWP and SWPB.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

## Accessing the ID\_ISAR0\_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_ISAR0_EL1	11	000	0000	0010	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.

# ID\_ISAR1\_EL1, AArch32 Instruction Set Attribute Register 1

The ID\_ISAR1\_EL1 characteristics are:

## Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID\\_ISAR0\\_EL1](#), [ID\\_ISAR2\\_EL1](#), [ID\\_ISAR3\\_EL1](#), [ID\\_ISAR4\\_EL1](#), and [ID\\_ISAR5\\_EL1](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register ID\_ISAR1\_EL1 is architecturally mapped to AArch32 System register [ID\\_ISAR1](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_ISAR1\_EL1 is a 32-bit register.

## Field descriptions

The ID\_ISAR1\_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
<a href="#">Jazelle</a>				<a href="#">Interwork</a>				<a href="#">Immediate</a>				<a href="#">IfThen</a>				<a href="#">Extend</a>				<a href="#">Except_AR</a>				<a href="#">Except</a>				<a href="#">Endian</a>			

### Jazelle, bits [31:28]

Indicates the implemented Jazelle extension instructions. Defined values are:

Jazelle	Meaning
0000	No support for Jazelle.
0001	Adds the BXJ instruction, and the J bit in the PSR. This setting might indicate a trivial implementation of the Jazelle extension.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Interwork, bits [27:24]

Indicates the implemented Interworking instructions. Defined values are:

Interwork	Meaning
0000	None implemented.
0001	Adds the BX instruction, and the T bit in the PSR.
0010	As for 0001, and adds the BLX instruction. PC loads have BX-like behavior.
0011	As for 0010, and guarantees that data-processing instructions in the A32 instruction set with the PC as the destination and the S bit clear have BX-like behavior.

All other values are reserved.

In ARMv8-A the only permitted value is 0011.

### Immediate, bits [23:20]

Indicates the implemented data-processing instructions with long immediates. Defined values are:

Immediate	Meaning
0000	None implemented.
0001	Adds: <ul style="list-style-type: none"> <li>• The MOVT instruction.</li> <li>• The MOV instruction encodings with zero-extended 16-bit immediates.</li> <li>• The T32 ADD and SUB instruction encodings with zero-extended 12-bit immediates, and the other ADD, ADR, and SUB encodings cross-referenced by the pseudocode for those encodings.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### IfThen, bits [19:16]

Indicates the implemented If-Then instructions in the T32 instruction set. Defined values are:

IfThen	Meaning
0000	None implemented.
0001	Adds the IT instructions, and the IT bits in the PSRs.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Extend, bits [15:12]

Indicates the implemented Extend instructions. Defined values are:

Extend	Meaning
0000	No scalar sign-extend or zero-extend instructions are implemented, where scalar instructions means non-Advanced SIMD instructions.
0001	Adds the SXTB, SXTB, UXTB, and UXTB instructions.
0010	As for 0001, and adds the SXTB16, SXTAB, SXTAB16, SXTAH, UXTB16, UXTAB, UXTAB16, and UXTAH instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

### Except\_AR, bits [11:8]

Indicates the implemented A and R profile exception-handling instructions. Defined values are:

Except_AR	Meaning
0000	None implemented.
0001	Adds the SRS and RFE instructions, and the A and R profile forms of the CPS instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Except, bits [7:4]

Indicates the implemented exception-handling instructions in the ARM instruction set. Defined values are:

Except	Meaning
0000	Not implemented. This indicates that the User bank and Exception return forms of the LDM and STM instructions are not implemented.
0001	Adds the LDM (exception return), LDM (user registers), and STM (user registers) instruction versions.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Endian, bits [3:0]

Indicates the implemented Endian instructions. Defined values are:

Endian	Meaning
0000	None implemented.
0001	Adds the SETEND instruction, and the E bit in the PSRs.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

## Accessing the ID\_ISAR1\_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_ISAR1_EL1	11	000	0000	0010	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<sup>®</sup> 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.



# ID\_ISAR2\_EL1, AArch32 Instruction Set Attribute Register 2

The ID\_ISAR2\_EL1 characteristics are:

## Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID\\_ISAR0\\_EL1](#), [ID\\_ISAR1\\_EL1](#), [ID\\_ISAR3\\_EL1](#), [ID\\_ISAR4\\_EL1](#), and [ID\\_ISAR5\\_EL1](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register ID\_ISAR2\_EL1 is architecturally mapped to AArch32 System register [ID\\_ISAR2](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_ISAR2\_EL1 is a 32-bit register.

## Field descriptions

The ID\_ISAR2\_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
<a href="#">Reversal</a>				<a href="#">PSR_AR</a>				<a href="#">MultU</a>				<a href="#">MultS</a>				<a href="#">Mult</a>				<a href="#">MultiAccessInt</a>				<a href="#">MemHint</a>				<a href="#">LoadStore</a>			

### Reversal, bits [31:28]

Indicates the implemented Reversal instructions. Defined values are:

Reversal	Meaning
0000	None implemented.
0001	Adds the REV, REV16, and REVSH instructions.
0010	As for 0001, and adds the RBIT instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

### PSR\_AR, bits [27:24]

Indicates the implemented A and R profile instructions to manipulate the PSR. Defined values are:

PSR_AR	Meaning
0000	None implemented.
0001	Adds the MRS and MSR instructions, and the exception return forms of data-processing instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

The exception return forms of the data-processing instructions are:



- In the A32 instruction set, data-processing instructions with the PC as the destination and the S bit set. These instructions might be affected by the WithShifts attribute.
- In the T32 instruction set, the SUBS PC,LR,#N instruction.

**MultU, bits [23:20]**

Indicates the implemented advanced unsigned Multiply instructions. Defined values are:

MultU	Meaning
0000	None implemented.
0001	Adds the UMULL and UMLAL instructions.
0010	As for 0001, and adds the UMAAL instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

**MultS, bits [19:16]**

Indicates the implemented advanced signed Multiply instructions. Defined values are:

MultS	Meaning
0000	None implemented.
0001	Adds the SMULL and SMLAL instructions.
0010	As for 0001, and adds the SMLABB, SMLABT, SMLALBB, SMLALBT, SMLALTB, SMLALTT, SMLATB, SMLATT, SMLAWB, SMLAWT, SMULBB, SMULBT, SMULTB, SMULTT, SMULWB, and SMULWT instructions. Also adds the Q bit in the PSRs.
0011	As for 0010, and adds the SMLAD, SMLADX, SMLALD, SMLALDX, SMLSD, SMLSDX, SMLSLD, SMLSLDX, SMMLA, SMMLAR, SMMLS, SMMLSR, SMMUL, SMMULR, SMUAD, SMUADX, SMUSD, and SMUSDX instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0011.

**Mult, bits [15:12]**

Indicates the implemented additional Multiply instructions. Defined values are:

Mult	Meaning
0000	No additional instructions implemented. This means only MUL is implemented.
0001	Adds the MLA instruction.
0010	As for 0001, and adds the MLS instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

**MultiAccessInt, bits [11:8]**

Indicates the support for interruptible multi-access instructions. Defined values are:

MultiAccessInt	Meaning
0000	No support. This means the LDM and STM instructions are not interruptible.
0001	LDM and STM instructions are restartable.
0010	LDM and STM instructions are continuable.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

**MemHint, bits [7:4]**

Indicates the implemented Memory Hint instructions. Defined values are:

MemHint	Meaning
0000	None implemented.
0001	Adds the PLD instruction.
0010	Adds the PLD instruction. (0001 and 0010 have identical effects.)
0011	As for 0001 (or 0010), and adds the PLI instruction.
0100	As for 0011, and adds the PLDW instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0100.

**LoadStore, bits [3:0]**

Indicates the implemented additional load/store instructions. Defined values are:

LoadStore	Meaning
0000	No additional load/store instructions implemented.
0001	Adds the LDRD and STRD instructions.
0010	As for 0001, and adds the Load Acquire (LDAB, LDAH, LDA, LDAEXB, LDAEXH, LDAEX, LDAEXD) and Store Release (STLB, STLH, STL, STLEXB, STLEXH, STLEX, STLEXD) instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

**Accessing the ID\_ISAR2\_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_ISAR2_EL1	11	000	0000	0010	010

**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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ID\_ISAR3\_EL1, AArch32 Instruction Set Attribute Register 3

The ID\_ISAR3\_EL1 characteristics are:

## Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID\\_ISAR0\\_EL1](#), [ID\\_ISAR1\\_EL1](#), [ID\\_ISAR2\\_EL1](#), [ID\\_ISAR4\\_EL1](#), and [ID\\_ISAR5\\_EL1](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register ID\_ISAR3\_EL1 is architecturally mapped to AArch32 System register [ID\\_ISAR3](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_ISAR3\_EL1 is a 32-bit register.

## Field descriptions

The ID\_ISAR3\_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
<a href="#">T32EE</a>				<a href="#">TrueNOP</a>				<a href="#">T32Copy</a>				<a href="#">TabBranch</a>				<a href="#">SynchPrim</a>				<a href="#">SVC</a>				<a href="#">SIMD</a>				<a href="#">Saturate</a>			

### T32EE, bits [31:28]

Indicates the implemented T32EE instructions. Defined values are:

T32EE	Meaning
0000	None implemented.
0001	Adds the ENTERX and LEAVEX instructions, and modifies the load behavior to include null checking.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### TrueNOP, bits [27:24]

Indicates the implemented true NOP instructions. Defined values are:

TrueNOP	Meaning
0000	None implemented. This means there are no NOP instructions that do not have any register dependencies.
0001	Adds true NOP instructions in both the T32 and A32 instruction sets. This also permits additional NOP-compatible hints.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**T32Copy, bits [23:20]**

Indicates the support for T32 non flag-setting MOV instructions. Defined values are:

<b>T32Copy</b>	<b>Meaning</b>
0000	Not supported. This means that in the T32 instruction set, encoding T1 of the MOV (register) instruction does not support a copy from a low register to a low register.
0001	Adds support for T32 instruction set encoding T1 of the MOV (register) instruction, copying from a low register to a low register.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**TabBranch, bits [19:16]**

Indicates the implemented Table Branch instructions in the T32 instruction set. Defined values are:

<b>TabBranch</b>	<b>Meaning</b>
0000	None implemented.
0001	Adds the TBB and TBH instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**SynchPrim, bits [15:12]**

Used in conjunction with ID\_ISAR4.SynchPrim\_frac to indicate the implemented Synchronization Primitive instructions. Defined values are:

<b>SynchPrim</b>	<b>Meaning</b>
0000	If SynchPrim_frac == 0000, no Synchronization Primitives implemented.
0001	If SynchPrim_frac == 0000, adds the LDREX and STREX instructions.
	If SynchPrim_frac == 0011, also adds the CLREX, LDREXB, STREXB, and STREXH instructions.
0010	If SynchPrim_frac == 0000, as for [0001, 0011] and also adds the LDREXD and STREXD instructions.

All other combinations of SynchPrim and SynchPrim\_frac are reserved.

In ARMv8-A the only permitted value is 0010.

**SVC, bits [11:8]**

Indicates the implemented SVC instructions. Defined values are:

<b>SVC</b>	<b>Meaning</b>
0000	Not implemented.
0001	Adds the SVC instruction.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**SIMD, bits [7:4]**

Indicates the implemented SIMD instructions. Defined values are:

SIMD	Meaning
0000	None implemented.
0001	Adds the SSAT and USAT instructions, and the Q bit in the PSRs.
0011	As for 0001, and adds the PKHBT, PKHTB, QADD16, QADD8, QASX, QSUB16, QSUB8, QSAX, SADD16, SADD8, SASX, SEL, SHADD16, SHADD8, SHASX, SHSUB16, SHSUB8, SHSAX, SSAT16, SSUB16, SSUB8, SSAX, SXTAB16, SXTB16, UADD16, UADD8, UASX, UHADD16, UHADD8, UHASX, UHSUB16, UHSUB8, UHSAX, UQADD16, UQADD8, UQASX, UQSUB16, UQSUB8, UQSAX, USAD8, USADA8, USAT16, USUB16, USUB8, USAX, UXTAB16, and UXTB16 instructions. Also adds support for the GE[3:0] bits in the PSRs.

All other values are reserved.

In ARMv8-A the only permitted value is 0011.

The SIMD field relates only to implemented instructions that perform SIMD operations on the general-purpose registers. In an implementation that supports floating-point and Advanced SIMD instructions, [MVFR0](#), [MVFR1](#), and [MVFR2](#) give information about the implemented Advanced SIMD instructions.

### Saturate, bits [3:0]

Indicates the implemented Saturate instructions. Defined values are:

Saturate	Meaning
0000	None implemented. This means no non-Advanced SIMD saturate instructions are implemented.
0001	Adds the QADD, QDADD, QDSUB, and QSUB instructions, and the Q bit in the PSRs.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

## Accessing the ID\_ISAR3\_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_ISAR3_EL1	11	000	0000	0010	011

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ID\_ISAR4\_EL1, AArch32 Instruction Set Attribute Register 4

The ID\_ISAR4\_EL1 characteristics are:

## Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID\\_ISAR0\\_EL1](#), [ID\\_ISAR1\\_EL1](#), [ID\\_ISAR2\\_EL1](#), [ID\\_ISAR3\\_EL1](#), and [ID\\_ISAR5\\_EL1](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register ID\_ISAR4\_EL1 is architecturally mapped to AArch32 System register [ID\\_ISAR4](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_ISAR4\_EL1 is a 32-bit register.

## Field descriptions

The ID\_ISAR4\_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
<a href="#">SWP_frac</a>				<a href="#">PSR_M</a>				<a href="#">SynchPrim_frac</a>				<a href="#">Barrier</a>				<a href="#">SMC</a>				<a href="#">Writeback</a>				<a href="#">WithShifts</a>				<a href="#">Unpriv</a>			

### SWP\_frac, bits [31:28]

Indicates support for the memory system locking the bus for SWP or SWPB instructions. Defined values are:

SWP_frac	Meaning
0000	SWP or SWPB instructions not implemented.
0001	SWP or SWPB implemented but only in a uniprocessor context. SWP and SWPB do not guarantee whether memory accesses from other masters can come between the load memory access and the store memory access of the SWP or SWPB.

All other values are reserved. This field is valid only if the [ID\\_ISAR0](#).Swap\_instrs field is 0000.

In ARMv8-A the only permitted value is 0000.

### PSR\_M, bits [27:24]

Indicates the implemented M profile instructions to modify the PSRs. Defined values are:

PSR_M	Meaning
0000	None implemented.
0001	Adds the M profile forms of the CPS, MRS, and MSR instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.



**SynchPrim\_frac, bits [23:20]**

Used in conjunction with [ID\\_ISAR3.SynchPrim](#) to indicate the implemented Synchronization Primitive instructions. Possible values are:

SynchPrim_frac	Meaning
0000	If SynchPrim == 0000, no Synchronization Primitives implemented. If SynchPrim == 0001, adds the LDREX and STREX instructions. If SynchPrim == 0010, also adds the CLREX, LDREXB, LDREXH, STREXB, STREXH, LDREXD, and STREXD instructions.
0011	If SynchPrim == 0001, adds the LDREX, STREX, CLREX, LDREXB, LDREXH, STREXB, and STREXH instructions.

All other combinations of SynchPrim and SynchPrim\_frac are reserved.

In ARMv8-A the only permitted value is 0000.

**Barrier, bits [19:16]**

Indicates the implemented Barrier instructions in the A32 and T32 instruction sets. Defined values are:

Barrier	Meaning
0000	None implemented. Barrier operations are provided only as System instructions in the (coproc==1111) encoding space.
0001	Adds the DMB, DSB, and ISB barrier instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**SMC, bits [15:12]**

Indicates the implemented SMC instructions. Defined values are:

SMC	Meaning
0000	None implemented.
0001	Adds the SMC instruction.

All other values are reserved.

In ARMv8-A the permitted values are 0001 and 0000.

If EL1 cannot use AArch32 then this field has the value 0000.

**Writeback, bits [11:8]**

Indicates the support for Writeback addressing modes. Defined values are:

Writeback	Meaning
0000	Basic support. Only the LDM, STM, PUSH, POP, SRS, and RFE instructions support writeback addressing modes. These instructions support all of their writeback addressing modes.
0001	Adds support for all of the writeback addressing modes.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**WithShifts, bits [7:4]**

Indicates the support for instructions with shifts. Defined values are:

WithShifts	Meaning
0000	Nonzero shifts supported only in MOV and shift instructions.
0001	Adds support for shifts of loads and stores over the range LSL 0-3.
0011	As for 0001, and adds support for other constant shift options, both on load/store and other instructions.
0100	As for 0011, and adds support for register-controlled shift options.

All other values are reserved.

In ARMv8-A the only permitted value is 0100.

### Unpriv, bits [3:0]

Indicates the implemented unprivileged instructions. Defined values are:

Unpriv	Meaning
0000	None implemented. No T variant instructions are implemented.
0001	Adds the LDRBT, LDRT, STRBT, and STRT instructions.
0010	As for 0001, and adds the LDRHT, LDRSBT, LDRSHT, and STRHT instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0010.

## Accessing the ID\_ISAR4\_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_ISAR4_EL1	11	000	0000	0010	100

## 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<sup>®</sup> 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.



# ID\_ISAR5\_EL1, AArch32 Instruction Set Attribute Register 5

The ID\_ISAR5\_EL1 characteristics are:

## Purpose

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with [ID\\_ISAR0\\_EL1](#), [ID\\_ISAR1\\_EL1](#), [ID\\_ISAR2\\_EL1](#), [ID\\_ISAR3\\_EL1](#), and [ID\\_ISAR4\\_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\_ISAR5\_EL1 is architecturally mapped to AArch32 System register [ID\\_ISAR5](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_ISAR5\_EL1 is a 32-bit register.

## Field descriptions

The ID\_ISAR5\_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		RDM			0	0	0	0		CRC32				SHA2				SHA1				AES				SEVL		

### Bits [31:28]

Reserved, RES0.

### RDM, bits [27:24]

In ARMv8.2 and ARMv8.1:

Indicates whether the VQRDMLAH and VQRDMLSH instructions are implemented in AArch32 state. Defined values are:

RDM	Meaning
0000	No VQRDMLAH and VQRDMLSH instructions implemented.
0001	VQRDMLAH and VQRDMLSH instructions implemented.

All other values are reserved.

In ARMv8.0 the only permitted value is 0000.

From ARMv8.1 the only permitted value is 0001. This feature is identified by the name ARMv8.1-RDMA.

### In ARMv8.0:

Reserved, RES0.

**Bits [23:20]**

Reserved, RES0.

**CRC32, bits [19:16]**

Indicates whether the CRC32 instructions are implemented in AArch32 state.

CRC32	Meaning
0000	No CRC32 instructions implemented.
0001	CRC32B, CRC32H, CRC32W, CRC32CB, CRC32CH, and CRC32CW instructions implemented.

All other values are reserved.

In ARMv8.0 the permitted values are 0000 and 0001.

From ARMv8.1 the only permitted value is 0001.

**SHA2, bits [15:12]**

Indicates whether the SHA2 instructions are implemented in AArch32 state.

SHA2	Meaning
0000	No SHA2 instructions implemented.
0001	SHA256H, SHA256H2, SHA256SU0, and SHA256SU1 implemented.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

**SHA1, bits [11:8]**

Indicates whether the SHA1 instructions are implemented in AArch32 state.

SHA1	Meaning
0000	No SHA1 instructions implemented.
0001	SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 implemented.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

**AES, bits [7:4]**

Indicates whether the AES instructions are implemented in AArch32 state.

AES	Meaning
0000	No AES instructions implemented.
0001	AESE, AESD, AESMC, and AESIMC implemented.
0010	As for 0001, plus PMULL/PMULL2 instructions operating on 64-bit data quantities.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0010.

**SEVL, bits [3:0]**

Indicates whether the SEVL instruction is implemented in AArch32 state.

SEVL	Meaning
0000	SEVL is implemented as a NOP.
0001	SEVL is implemented as Send Event Local.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

## Accessing the ID\_ISAR5\_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_ISAR5_EL1	11	000	0000	0010	101

## 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.

# ID\_MMFR0\_EL1, AArch32 Memory Model Feature Register 0

The ID\_MMFR0\_EL1 characteristics are:

## Purpose

Provides information about the implemented memory model and memory management support in AArch32 state.

Must be interpreted with [ID\\_MMFR1\\_EL1](#), [ID\\_MMFR2\\_EL1](#), [ID\\_MMFR3\\_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\_MMFR0\_EL1 is architecturally mapped to AArch32 System register [ID\\_MMFR0](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_MMFR0\_EL1 is a 32-bit register.

## Field descriptions

The ID\_MMFR0\_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
<a href="#">InnerShr</a>				<a href="#">FCSE</a>				<a href="#">AuxReg</a>				<a href="#">TCM</a>				<a href="#">ShareLvl</a>				<a href="#">OuterShr</a>				<a href="#">PMSA</a>				<a href="#">VMSA</a>			

### InnerShr, bits [31:28]

Innermost Shareability. Indicates the innermost shareability domain implemented. Defined values are:

InnerShr	Meaning
0000	Implemented as Non-cacheable.
0001	Implemented with hardware coherency support.
1111	Shareability ignored.

All other values are reserved.

In ARMv8 the permitted values are 0000, 0001, and 1111.

This field is valid only if the implementation supports two levels of shareability, as indicated by ID\_MMFR0\_EL1.ShareLvl having the value 0001.

When ID\_MMFR0\_EL1.ShareLvl is zero, this field is UNK.

### FCSE, bits [27:24]

Indicates whether the implementation includes the FCSE. Defined values are:

FCSE	Meaning
0000	Not supported.
0001	Support for FCSE.

All other values are reserved.

In ARMv8 the only permitted value is 0000.

### AuxReg, bits [23:20]

Auxiliary Registers. Indicates support for Auxiliary registers. Defined values are:

AuxReg	Meaning
0000	None supported.
0001	Support for Auxiliary Control Register only.
0010	Support for Auxiliary Fault Status Registers ( <a href="#">AIFSR</a> and <a href="#">ADEFSR</a> ) and Auxiliary Control Register.

All other values are reserved.

In ARMv8 the only permitted value is 0010.

#### Note

Accesses to unimplemented Auxiliary registers are UNDEFINED.

### TCM, bits [19:16]

Indicates support for TCMs and associated DMAs. Defined values are:

TCM	Meaning
0000	Not supported.
0001	Support is IMPLEMENTATION DEFINED. ARMv7 requires this setting.
0010	Support for TCM only, ARMv6 implementation.
0011	Support for TCM and DMA, ARMv6 implementation.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### ShareLvl, bits [15:12]

Shareability Levels. Indicates the number of shareability levels implemented. Defined values are:

ShareLvl	Meaning
0000	One level of shareability implemented.
0001	Two levels of shareability implemented.

All other values are reserved.

In ARMv8 the only permitted value is 0001.

### OuterShr, bits [11:8]

Outermost Shareability. Indicates the outermost shareability domain implemented. Defined values are:

OuterShr	Meaning
0000	Implemented as Non-cacheable.
0001	Implemented with hardware coherency support.
1111	Shareability ignored.

All other values are reserved.

In ARMv8 the permitted values are 0000, 0001, and 1111.

### PMSA, bits [7:4]

Indicates support for a PMSA. Defined values are:



PMSA	Meaning
0000	Not supported.
0001	Support for IMPLEMENTATION DEFINED PMSA.
0010	Support for PMSAv6, with a Cache Type Register implemented.
0011	Support for PMSAv7, with support for memory subsections. ARMv7-R profile.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### VMSA, bits [3:0]

Indicates support for a VMSA. Defined values are:

VMSA	Meaning
0000	Not supported.
0001	Support for IMPLEMENTATION DEFINED VMSA.
0010	Support for VMSAv6, with Cache and TLB Type Registers implemented.
0011	Support for VMSAv7, with support for remapping and the Access flag. ARMv7-A profile.
0100	As for 0011, and adds support for the PXN bit in the Short-descriptor translation table format descriptors.
0101	As for 0100, and adds support for the Long-descriptor translation table format.

All other values are reserved.

In ARMv8-A the only permitted value is 0101.

## Accessing the ID\_MMFR0\_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_MMFR0_EL1	11	000	0000	0001	100

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ID\_MMFR1\_EL1, AArch32 Memory Model Feature Register 1

The ID\_MMFR1\_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\\_MMFR2\\_EL1](#), [ID\\_MMFR3\\_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\_MMFR1\_EL1 is architecturally mapped to AArch32 System register [ID\\_MMFR1](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_MMFR1\_EL1 is a 32-bit register.

## Field descriptions

The ID\_MMFR1\_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
BPred				L1TstCln				L1Uni				L1Hvd				L1UniSW				L1HvdSW				L1UniVA				L1HvdVA			

### BPred, bits [31:28]

Branch Predictor. Indicates branch predictor management requirements. Defined values are:

BPred	Meaning
0000	No branch predictor, or no MMU present. Implies a fixed MPU configuration.
0001	Branch predictor requires flushing on: <ul style="list-style-type: none"> <li>Enabling or disabling a stage of address translation.</li> <li>Writing new data to instruction locations.</li> <li>Writing new mappings to the translation tables.</li> <li>Changes to the <a href="#">TTBR0</a>, <a href="#">TTBR1</a>, or <a href="#">TTBCR</a> registers.</li> <li>Changes to the ContextID or ASID, or to the FCSE ProcessID if this is supported.</li> </ul>
0010	Branch predictor requires flushing on: <ul style="list-style-type: none"> <li>Enabling or disabling a stage of address translation.</li> <li>Writing new data to instruction locations.</li> <li>Writing new mappings to the translation tables.</li> <li>Any change to the <a href="#">TTBR0</a>, <a href="#">TTBR1</a>, or <a href="#">TTBCR</a> registers without a change to the corresponding ContextID or ASID, or FCSE ProcessID if this is supported.</li> </ul>
0011	Branch predictor requires flushing only on writing new data to instruction locations.
0100	For execution correctness, branch predictor requires no flushing at any time.

All other values are reserved.

In ARMv8-A the permitted values are 0010, 0011, or 0100. For values other than 0000 and 0100 the ARM Architecture Reference Manual, or the product documentation, might give more information about the required maintenance.

**L1TstCln, bits [27:24]**

Level 1 cache Test and Clean. Indicates the supported Level 1 data cache test and clean operations, for Harvard or unified cache implementations. Defined values are:

<b>L1TstCln</b>	<b>Meaning</b>
0000	None supported.
0001	Supported Level 1 data cache test and clean operations are: <ul style="list-style-type: none"> <li>• Test and clean data cache.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Test, clean, and invalidate data cache.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

**L1Uni, bits [23:20]**

Level 1 Unified cache. Indicates the supported entire Level 1 cache maintenance operations for a unified cache implementation. Defined values are:

<b>L1Uni</b>	<b>Meaning</b>
0000	None supported.
0001	Supported entire Level 1 cache operations are: <ul style="list-style-type: none"> <li>• Invalidate cache, including branch predictor if appropriate.</li> <li>• Invalidate branch predictor, if appropriate.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Clean cache, using a recursive model that uses the cache dirty status bit.</li> <li>• Clean and invalidate cache, using a recursive model that uses the cache dirty status bit.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

**L1Hvd, bits [19:16]**

Level 1 Harvard cache. Indicates the supported entire Level 1 cache maintenance operations for a Harvard cache implementation. Defined values are:

<b>L1Hvd</b>	<b>Meaning</b>
0000	None supported.
0001	Supported entire Level 1 cache operations are: <ul style="list-style-type: none"> <li>• Invalidate instruction cache, including branch predictor if appropriate.</li> <li>• Invalidate branch predictor, if appropriate.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Invalidate data cache.</li> <li>• Invalidate data cache and instruction cache, including branch predictor if appropriate.</li> </ul>
0011	As for 0010, and adds: <ul style="list-style-type: none"> <li>• Clean data cache, using a recursive model that uses the cache dirty status bit.</li> <li>• Clean and invalidate data cache, using a recursive model that uses the cache dirty status bit.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

**L1UniSW, bits [15:12]**

Level 1 Unified cache by Set/Way. Indicates the supported Level 1 cache line maintenance operations by set/way, for a unified cache implementation. Defined values are:

<b>L1UniSW</b>	<b>Meaning</b>
0000	None supported.
0001	Supported Level 1 unified cache line maintenance operations by set/way are: <ul style="list-style-type: none"> <li>• Clean cache line by set/way.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Clean and invalidate cache line by set/way.</li> </ul>
0011	As for 0010, and adds: <ul style="list-style-type: none"> <li>• Invalidate cache line by set/way.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

#### **L1HvdSW, bits [11:8]**

Level 1 Harvard cache by Set/Way. Indicates the supported Level 1 cache line maintenance operations by set/way, for a Harvard cache implementation. Defined values are:

<b>L1HvdSW</b>	<b>Meaning</b>
0000	None supported.
0001	Supported Level 1 Harvard cache line maintenance operations by set/way are: <ul style="list-style-type: none"> <li>• Clean data cache line by set/way.</li> <li>• Clean and invalidate data cache line by set/way.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Invalidate data cache line by set/way.</li> </ul>
0011	As for 0010, and adds: <ul style="list-style-type: none"> <li>• Invalidate instruction cache line by set/way.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

#### **L1UniVA, bits [7:4]**

Level 1 Unified cache by Virtual Address. Indicates the supported Level 1 cache line maintenance operations by VA, for a unified cache implementation. Defined values are:

<b>L1UniVA</b>	<b>Meaning</b>
0000	None supported.
0001	Supported Level 1 unified cache line maintenance operations by VA are: <ul style="list-style-type: none"> <li>• Clean cache line by VA.</li> <li>• Invalidate cache line by VA.</li> <li>• Clean and invalidate cache line by VA.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Invalidate branch predictor by VA, if branch predictor is implemented.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

#### **L1HvdVA, bits [3:0]**

Level 1 Harvard cache by Virtual Address. Indicates the supported Level 1 cache line maintenance operations by VA, for a Harvard cache implementation. Defined values are:

<b>L1HvdVA</b>	<b>Meaning</b>
0000	None supported.
0001	Supported Level 1 Harvard cache line maintenance operations by VA are: <ul style="list-style-type: none"> <li>• Clean data cache line by VA.</li> <li>• Invalidate data cache line by VA.</li> <li>• Clean and invalidate data cache line by VA.</li> <li>• Clean instruction cache line by VA.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Invalidate branch predictor by VA, if branch predictor is implemented.</li> </ul>

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

## Accessing the ID\_MMFR1\_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_MMFR1_EL1	11	000	0000	0001	101

## 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.

# ID\_MMFR2\_EL1, AArch32 Memory Model Feature Register 2

The ID\_MMFR2\_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\\_MMFR3\\_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\_MMFR2\_EL1 is architecturally mapped to AArch32 System register [ID\\_MMFR2](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_MMFR2\_EL1 is a 32-bit register.

## Field descriptions

The ID\_MMFR2\_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
<a href="#">HWAaccFlg</a>				<a href="#">WFIStall</a>				<a href="#">MemBarr</a>				<a href="#">UniTLB</a>				<a href="#">HvdTLB</a>				<a href="#">L1HvdRng</a>				<a href="#">L1HvdBG</a>				<a href="#">L1HvdFG</a>			

### HWAaccFlg, bits [31:28]

Hardware Access Flag. In earlier versions of the ARM Architecture, this field indicates support for a Hardware Access flag, as part of the VMSAv7 implementation. Defined values are:

HWAaccFlg	Meaning
0000	Not supported.
0001	Support for VMSAv7 Access flag, updated in hardware.

All other values are reserved.

In ARMv8 the only permitted value is 0000.

### WFIStall, bits [27:24]

Wait For Interrupt Stall. Indicates the support for Wait For Interrupt (WFI) stalling. Defined values are:

WFIStall	Meaning
0000	Not supported.
0001	Support for WFI stalling.

All other values are reserved.

In ARMv8 the permitted values are 0000 and 0001.

**MemBarr, bits [23:20]**

Memory Barrier. Indicates the supported memory barrier System instructions in the (coproc==1111) encoding space:

MemBarr	Meaning
0000	None supported.
0001	Supported memory barrier System instructions are: <ul style="list-style-type: none"> <li>• Data Synchronization Barrier (DSB).</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Instruction Synchronization Barrier (ISB).</li> <li>• Data Memory Barrier (DMB).</li> </ul>

All other values are reserved.

In ARMv8 the only permitted value is 0010.

ARM deprecates the use of these operations. ID\_ISAR4.Barrier\_instrs indicates the level of support for the preferred barrier instructions.

**UniTLB, bits [19:16]**

Unified TLB. Indicates the supported TLB maintenance operations, for a unified TLB implementation. Defined values are:

UniTLB	Meaning
0000	Not supported.
0001	Supported unified TLB maintenance operations are: <ul style="list-style-type: none"> <li>• Invalidate all entries in the TLB.</li> <li>• Invalidate TLB entry by VA.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Invalidate TLB entries by ASID match.</li> </ul>
0011	As for 0010, and adds: <ul style="list-style-type: none"> <li>• Invalidate instruction TLB and data TLB entries by VA All ASID. This is a shared unified TLB operation.</li> </ul>
0100	As for 0011, and adds: <ul style="list-style-type: none"> <li>• Invalidate Hyp mode unified TLB entry by VA.</li> <li>• Invalidate entire Non-secure PL1&amp;0 unified TLB.</li> <li>• Invalidate entire Hyp mode unified TLB.</li> </ul>
0101	As for 0100, and adds the following operations: <a href="#">TLBIMVALIS</a> , <a href="#">TLBIMVAALIS</a> , <a href="#">TLBIMVALHIS</a> , <a href="#">TLBIMVAL</a> , <a href="#">TLBIMVAAL</a> , <a href="#">TLBIMVALH</a> .
0110	As for 0101, and adds the following operations: <a href="#">TLBIIPAS2IS</a> , <a href="#">TLBIIPAS2LIS</a> , <a href="#">TLBIIPAS2</a> , <a href="#">TLBIIPAS2L</a> .

All other values are reserved.

In ARMv8-A the only permitted value is 0110.

**HvdTLB, bits [15:12]**

If the Unified TLB field (UniTLB, bits [19:16]) is not 0000, then the meaning of this field is IMPLEMENTATION DEFINED. ARM deprecates the use of this field by software.

**L1HvdRng, bits [11:8]**

Level 1 Harvard cache Range. Indicates the supported Level 1 cache maintenance range operations, for a Harvard cache implementation. Defined values are:

L1HvdRng	Meaning
0000	Not supported.
0001	Supported Level 1 Harvard cache maintenance range operations are: <ul style="list-style-type: none"> <li>• Invalidate data cache range by VA.</li> <li>• Invalidate instruction cache range by VA.</li> <li>• Clean data cache range by VA.</li> <li>• Clean and invalidate data cache range by VA.</li> </ul>

All other values are reserved.

In ARMv8 the only permitted value is 0000.



**L1HvdBG, bits [7:4]**

Level 1 Harvard cache Background fetch. Indicates the supported Level 1 cache background fetch operations, for a Harvard cache implementation. When supported, background fetch operations are non-blocking operations. Defined values are:

L1HvdBG	Meaning
0000	Not supported.
0001	Supported Level 1 Harvard cache background fetch operations are: <ul style="list-style-type: none"> <li>Fetch instruction cache range by VA.</li> <li>Fetch data cache range by VA.</li> </ul>

All other values are reserved.

In ARMv8 the only permitted value is 0000.

**L1HvdFG, bits [3:0]**

Level 1 Harvard cache Foreground fetch. Indicates the supported Level 1 cache foreground fetch operations, for a Harvard cache implementation. When supported, foreground fetch operations are blocking operations. Defined values are:

L1HvdFG	Meaning
0000	Not supported.
0001	Supported Level 1 Harvard cache foreground fetch operations are: <ul style="list-style-type: none"> <li>Fetch instruction cache range by VA.</li> <li>Fetch data cache range by VA.</li> </ul>

All other values are reserved.

In ARMv8 the only permitted value is 0000.

**Accessing the ID\_MMFR2\_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_MMFR2_EL1	11	000	0000	0001	110

**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<sup>®</sup> 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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
<a href="#">Supersec</a>				<a href="#">CMemSz</a>				<a href="#">CohWalk</a>				<a href="#">PAN</a>				<a href="#">MaintBcst</a>				<a href="#">BPMaint</a>				<a href="#">CMaintSW</a>				<a href="#">CMaintVA</a>			

### 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 ARMv8.2 and ARMv8.1:

Privileged Access Never. Indicates support for the PAN bit in [CPSR](#), [SPSR](#), and [DPSR](#) in AArch32. Defined values are:

PAN	Meaning
0000	PAN not supported.
0001	PAN supported.
0010	PAN supported and <a href="#">ATS1CPRP</a> and <a href="#">ATS1CPWP</a> instructions supported.

All other values are reserved.

In ARMv8.0 the only permitted value is 0000.

In ARMv8.1 the only permitted value is 0001. This feature is identified by the name ARMv8.1-PAN.

From ARMv8.2, the only permitted value is 0010. This feature is identified by the name ARMv8.2-ATS1E1.

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"> <li>• Invalidate all branch predictors.</li> </ul>
0010	As for 0001, and adds: <ul style="list-style-type: none"> <li>• Invalidate branch predictors by VA.</li> </ul>

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"> <li>• Invalidate data cache by set/way.</li> <li>• Clean data cache by set/way.</li> <li>• Clean and invalidate data cache by set/way.</li> </ul>

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"> <li>• Invalidate data cache by VA.</li> <li>• Clean data cache by VA.</li> <li>• Clean and invalidate data cache by VA.</li> <li>• Invalidate instruction cache by VA.</li> <li>• Invalidate all instruction cache entries.</li> </ul>

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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- If [HCR\\_EL2.TID3](#)=1, Non-secure read accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ID\_MMFR4\_EL1, AArch32 Memory Model Feature Register 4

The ID\_MMFR4\_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\\_MMFR3\\_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\_MMFR4\_EL1 is architecturally mapped to AArch32 System register [ID\\_MMFR4](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_MMFR4\_EL1 is a 32-bit register.

## Field descriptions

The ID\_MMFR4\_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		LSM				HPDS				CnP				XNX				AC2				SpecSEI		

### Bits [31:24]

Reserved, RAZ.

### LSM, bits [23:20]

In ARMv8.2:

Indicates support for LSMAOE and nTLSMD bits in [HSCTLR](#) and [SCTLR](#). Defined values are:

LSM	Meaning
0000	LSMAOE and nTLSMD bits not supported.
0001	LSMAOE and nTLSMD bits supported.

All other values are reserved.

In ARMv8.0 and ARMv8.1 the only permitted value is 0000.

From ARMv8.2, the permitted values are 0000 and 0001. This feature is identified by the name ARMv8.2-LSMAOC.

In ARMv8.1 and ARMv8.0:

Reserved, RAZ.

**HPDS, bits [19:16]****In ARMv8.2:**

Hierarchical permission disables bits in translation tables. Defined values are:

HPDS	Meaning
0000	Disabling of hierarchical controls not supported.
0001	Supports disabling of hierarchical controls using the <a href="#">TTBCR2</a> .HPD0, <a href="#">TTBCR2</a> .HPD1, and <a href="#">HTCR</a> .HPD bits.
0010	Supports disabling of hierarchical controls using the <a href="#">TTBCR2</a> .HPD0, <a href="#">TTBCR2</a> .HPD1, and <a href="#">HTCR</a> .HPD bits, and hardware allocation of bits[62:59] of the last level page table descriptor for IMPLEMENTATION DEFINED use.

All other values are reserved.

In ARMv8.0 and ARMv8.1 the only permitted value is 0000.

From ARMv8.2, the permitted values are 0000, 0001 and 0010. This feature is identified by the name ARMv8.2-AA32HPD.

**Note**

The encoding 0000 implies that the encoding for [TTBCR2](#) is unallocated.

**In ARMv8.1 and ARMv8.0:**

Reserved, RAZ.

**CnP, bits [15:12]****In ARMv8.2:**

Common not Private translations. Defined values are:

CnP	Meaning
0000	Common not Private translations not supported.
0001	Common not Private translations supported.

All other values are reserved.

In ARMv8.0 and ARMv8.1 the only permitted value is 0000.

From ARMv8.2, the only permitted value is 0001. This feature is identified by the name ARMv8.2-TTCNP.

**In ARMv8.1 and ARMv8.0:**

Reserved, RAZ.

**XNX, bits [11:8]****In ARMv8.2:**

Support for execute never control distinction at stage 2 bit. Defined values are:

XNX	Meaning
0000	Distinction between EL0 and EL1 execute permission at stage 2 not supported.
0001	Distinction between EL0 and EL1 execute permission at stage 2 supported.

All other values are reserved.

In ARMv8.0 and ARMv8.1 the only permitted value is 0000.

From ARMv8.2, the only permitted value is 0001. This feature is identified by the name ARMv8.2-TTS2UXN.



**In ARMv8.1 and ARMv8.0:**

Reserved, RAZ.

**AC2, bits [7:4]**

Indicates the extension of the [ACTLR](#) and [HACTLR](#) registers using [ACTLR2](#) and [HACTLR2](#). Defined values are:

AC2	Meaning
0000	<a href="#">ACTLR2</a> and <a href="#">HACTLR2</a> are not implemented.
0001	<a href="#">ACTLR2</a> and <a href="#">HACTLR2</a> are implemented.

All other values are reserved.

In ARMv8.0 and ARMv8.1 the permitted values are 0000 and 0001.

From ARMv8.2, the only permitted value is 0001.

**SpecSEI, bits [3:0]**

Describes whether the PE can generate SError interrupt exceptions from speculative reads of memory, including speculative instruction fetches. The defined values of this field are:

SpecSEI	Meaning
0000	The PE never generates an SError interrupt due to an external abort on a speculative read.
0001	The PE might generate an SError interrupt due to an external abort on a speculative read.

All other values are reserved.

When the RAS Extension is not implemented, this field is RAZ.

**Accessing the ID\_MMFR4\_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_MMFR4_EL1	11	000	0000	0010	110

**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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- If [HCR\\_EL2.TID3](#)=1, Non-secure read accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ID\_PFR0\_EL1, AArch32 Processor Feature Register 0

The ID\_PFR0\_EL1 characteristics are:

## Purpose

Gives top-level information about the instruction sets supported by the PE in AArch32 state.

Must be interpreted with [ID\\_PFR1\\_EL1](#).

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section D7.1.3.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register ID\_PFR0\_EL1 is architecturally mapped to AArch32 System register [ID\\_PFR0](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_PFR0\_EL1 is a 32-bit register.

## Field descriptions

The ID\_PFR0\_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
<a href="#">RAS</a>				0	0	0	0	0	0	0	0	0	0	0	0	<a href="#">State3</a>				<a href="#">State2</a>				<a href="#">State1</a>				<a href="#">State0</a>			

### 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.

### Bits [27:16]

Reserved, RES0.

### State3, bits [15:12]

T32EE instruction set support. Defined values are:

State3	Meaning
0000	Not implemented.
0001	T32EE instruction set implemented.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

**State2, bits [11:8]**

Jazelle extension support. Defined values are:

State2	Meaning
0000	Not implemented.
0001	Jazelle extension implemented, without clearing of <a href="#">JOSCR.CV</a> on exception entry.
0010	Jazelle extension implemented, with clearing of <a href="#">JOSCR.CV</a> on exception entry.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**State1, bits [7:4]**

T32 instruction set support. Defined values are:

State1	Meaning
0000	T32 instruction set not implemented.
0001	T32 encodings before the introduction of Thumb-2 technology implemented: <ul style="list-style-type: none"> <li>• All instructions are 16-bit.</li> <li>• A BL or BLX is a pair of 16-bit instructions.</li> <li>• 32-bit instructions other than BL and BLX cannot be encoded.</li> </ul>
0011	T32 encodings after the introduction of Thumb-2 technology implemented, for all 16-bit and 32-bit T32 basic instructions.

All other values are reserved.

In ARMv8-A the only permitted value is 0011.

**State0, bits [3:0]**

A32 instruction set support. Defined values are:

State0	Meaning
0000	A32 instruction set not implemented.
0001	A32 instruction set implemented.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

**Accessing the ID\_PFR0\_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_PFR0_EL1	11	000	0000	0001	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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- If [HCR\\_EL2.TID3](#)=1, Non-secure read accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ID\_PFR1\_EL1, AArch32 Processor Feature Register 1

The ID\_PFR1\_EL1 characteristics are:

## Purpose

Gives information about the AArch32 programmers' model.

Must be interpreted with [ID\\_PFR0\\_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\_PFR1\_EL1 is architecturally mapped to AArch32 System register [ID\\_PFR1](#).

In an implementation that supports only AArch64 state, this register is UNKNOWN.

## Attributes

ID\_PFR1\_EL1 is a 32-bit register.

## Field descriptions

The ID\_PFR1\_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
<a href="#">GIC</a>				<a href="#">Virt_frac</a>				<a href="#">Sec_frac</a>				<a href="#">GenTimer</a>				<a href="#">Virtualization</a>				<a href="#">MProgMod</a>				<a href="#">Security</a>				<a href="#">ProgMod</a>			

### GIC, bits [31:28]

System register GIC CPU interface. Defined values are:

GIC	Meaning
0000	No System register interface to the GIC CPU interface 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.

### Virt\_frac, bits [27:24]

Virtualization fractional field. When the Virtualization field is 0000, determines the support for features from the ARMv7 Virtualization Extensions. Defined values are:

Virt_frac	Meaning
0000	No features from the ARMv7 Virtualization Extensions are implemented.
0001	The following features of the ARMv7 Virtualization Extensions are implemented: <ul style="list-style-type: none"> <li>The <a href="#">SCR</a>.SIF bit, if EL3 is implemented.</li> <li>The modifications to the <a href="#">SCR</a>.AW and <a href="#">SCR</a>.FW bits described in the Virtualization Extensions, if EL3 is implemented.</li> <li>The MSR (Banked register) and MRS (Banked register) instructions.</li> <li>The ERET instruction.</li> </ul>

All other values are reserved.

In ARMv8-A the permitted values are:

- 0000 when EL2 is implemented.
- 0001 when EL2 is not implemented.

This field is only valid when the value of ID\_PFR1\_EL1.Virtualization is 0, otherwise it holds the value 0000.

---

#### Note

The ID\_ISAR registers do not identify whether the instructions added by the ARMv7 Virtualization Extensions are implemented.

---

### Sec\_frac, bits [23:20]

Security fractional field. When the Security field is 0000, determines the support for features from the ARMv7 Security Extensions. Defined values are:

Sec_frac	Meaning
0000	No features from the ARMv7 Security Extensions are implemented.
0001	The following features from the ARMv7 Security Extensions are implemented: <ul style="list-style-type: none"> <li>• The VBAR register.</li> <li>• The <a href="#">TTBCR</a>.PD0 and <a href="#">TTBCR</a>.PD1 bits.</li> </ul>
0010	As for 0001, plus the ability to access Secure or Non-secure physical memory is supported.

All other values are reserved.

In ARMv8-A the permitted values are:

- 0000 when EL3 is implemented.
- 0001 or 0010 when EL3 is not implemented.

This field is only valid when the value of ID\_PFR1\_EL1.Security is 0, otherwise it holds the value 0000.

### GenTimer, bits [19:16]

Generic Timer support. Defined values are:

GenTimer	Meaning
0000	Not implemented.
0001	Generic Timer implemented.

All other values are reserved.

In ARMv8-A the only permitted value is 0001.

### Virtualization, bits [15:12]

Virtualization support. Defined values are:

Virtualization	Meaning
0000	EL2, Hyp mode, and the HVC instruction not implemented.
0001	EL2, Hyp mode, the HVC instruction, and all the features described by Virt_frac == 0001 implemented.

All other values are reserved.

In ARMv8-A the permitted values are:

- 0000 when EL2 is not implemented.
- 0001 when EL2 is implemented.

In an implementation that includes EL2, if EL2 cannot use AArch32 but EL1 can use AArch32 then this field has the value 0001.

If EL1 cannot use AArch32 then this field has the value 0000.

**Note**

The ID\_ISARs do not identify whether the HVC instruction is implemented.

**MProgMod, bits [11:8]**

M profile programmers' model support. Defined values are:

MProgMod	Meaning
0000	Not supported.
0010	Support for two-stack programmers' model.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

**Security, bits [7:4]**

Security support. Defined values are:

Security	Meaning
0000	EL3, Monitor mode, and the SMC instruction not implemented.
0001	EL3, Monitor mode, the SMC instruction, and all the features described by Sec_frac == 0001 implemented.
0010	As for 0001, and adds the ability to set the <a href="#">NSACR</a> .RFR bit. Not permitted in ARMv8 as the <a href="#">NSACR</a> .RFR bit is RES0.

All other values are reserved.

In ARMv8-A the permitted values are:

- 0000 when EL3 is not implemented.
- 0001 when EL3 is implemented.

In an implementation that includes EL3, if EL3 cannot use AArch32 but EL1 can use AArch32 then this field has the value 0001.

If EL1 cannot use AArch32 then this field has the value 0000.

**ProgMod, bits [3:0]**

Support for the standard programmers' model for ARMv4 and later. Model must support User, FIQ, IRQ, Supervisor, Abort, Undefined, and System modes. Defined values are:

ProgMod	Meaning
0000	Not supported.
0001	Supported.

All other values are reserved.

In ARMv8-A the permitted values are 0001 and 0000.

If EL1 cannot use AArch32 then this field has the value 0000.

**Accessing the ID\_PFR1\_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_PFR1_EL1	11	000	0000	0001	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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=0$  :

- 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  $\text{SCR\_EL3.NS}=1$  &&  $\text{HCR\_EL2.E2H}=1$  &&  $\text{HCR\_EL2.TGE}=0$  :

- If [HCR\\_EL2.TID3](#)=1, Non-secure read accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# IFSR32\_EL2, Instruction Fault Status Register (EL2)

The IFSR32\_EL2 characteristics are:

## Purpose

Allows access to the AArch32 [IFSR](#) register from AArch64 state only. Its value has no effect on execution in AArch64 state.

This register is part of the Exception and fault handling registers functional group.

## Configuration

AArch64 System register IFSR32\_EL2 is architecturally mapped to AArch32 System register [IFSR](#).

If EL1 is AArch64 only, this register is UNDEFINED.

If EL2 is not implemented but EL3 is implemented, and EL1 is capable of using AArch32, then this register is not RES0.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

IFSR32\_EL2 is a 32-bit register.

## Field descriptions

The IFSR32\_EL2 bit assignments are:

### When TTBCR.EAE==0:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FnV	0	0	0	Ext	0	FS[4]	LPAE	0	0	0	0	0	FS[3:0]			

### Bits [31:17]

Reserved, RES0.

### FnV, bit [16]

FAR not Valid, for a Synchronous external abort other than a Synchronous external abort on a translation table walk.

FnV	Meaning
0	<a href="#">IFAR</a> is valid.
1	<a href="#">IFAR</a> is not valid, and holds an UNKNOWN value.

This field is only valid for a Synchronous external abort other than a Synchronous external abort on a translation table walk. It is RES0 for all other Prefetch Abort exceptions.

### Bits [15:13]

Reserved, RES0.

### ExT, bit [12]

External abort type. This bit can be used to provide an IMPLEMENTATION DEFINED classification of external aborts.

In an implementation that does not provide any classification of external aborts, this bit is RES0.

For aborts other than external aborts this bit always returns 0.

#### Bit [11]

Reserved, RES0.

#### FS[4], bit [10]

See FS[3:0], bits [3:0] for description of the FS field.

#### LPAE, bit [9]

On taking a Data Abort exception, this bit is set as follows:

LPAE	Meaning
0	Using the Short-descriptor translation table formats.
1	Using the Long-descriptor translation table formats.

Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.

#### Bits [8:4]

Reserved, RES0.

#### FS[3:0], bits [3:0]

Fault status bits. Interpreted with bit [10]. Possible values of FS[4:0] are:

FS	Meaning
00001	PC alignment fault
00010	Debug exception
00011	Access flag fault, level 1
00101	Translation fault, level 1
00110	Access flag fault, level 2
00111	Translation fault, level 2
01000	Synchronous external abort, not on translation table walk
01001	Domain fault, level 1
01011	Domain fault, level 2
01100	Synchronous external abort, on translation table walk, level 1
01101	Permission fault, level 1
01110	Synchronous external abort, on translation table walk, level 2
01111	Permission fault, level 2
10000	TLB conflict abort
10100	IMPLEMENTATION DEFINED fault (Lockdown fault)
11001	Synchronous parity or ECC error on memory access, not on translation table walk
11100	Synchronous parity or ECC error on translation table walk, level 1
11110	Synchronous parity or ECC error on translation table walk, level 2

All other values are reserved.

When the RAS Extension is implemented, 11001, 11100, and 11110, are reserved.

#### When TTBCR.EAE==1:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FnV	0	0	0	Ext	0	0	LPAE	0	0	0	STATUS					

**Bits [31:17]**

Reserved, RES0.

**FnV, bit [16]**

FAR not Valid, for a Synchronous external abort other than a Synchronous external abort on a translation table walk.

FnV	Meaning
0	<a href="#">IFAR</a> is valid.
1	<a href="#">IFAR</a> is not valid, and holds an UNKNOWN value.

This field is only valid for a Synchronous external abort other than a Synchronous external abort on a translation table walk. It is RES0 for all other Prefetch Abort exceptions.

**Bits [15:13]**

Reserved, RES0.

**ExT, bit [12]**

External abort type. This bit can be used to provide an IMPLEMENTATION DEFINED classification of external aborts.

In an implementation that does not provide any classification of external aborts, this bit is RES0.

For aborts other than external aborts this bit always returns 0.

**Bits [11:10]**

Reserved, RES0.

**LPAE, bit [9]**

On taking a Data Abort exception, this bit is set as follows:

LPAE	Meaning
0	Using the Short-descriptor translation table formats.
1	Using the Long-descriptor translation table formats.

Hardware does not interpret this bit to determine the behavior of the memory system, and therefore software can set this bit to 0 or 1 without affecting operation.

**Bits [8:6]**

Reserved, RES0.

**STATUS, bits [5:0]**

Fault status bits. All encodings not shown below are reserved:

STATUS	Meaning
000000	Address size fault in <a href="#">TTBR0</a> or <a href="#">TTBR1</a>
000001	Address size fault, level 1
000010	Address size fault, level 2
000011	Address size fault, level 3
000101	Translation fault, level 1
000110	Translation fault, level 2
000111	Translation fault, level 3
001001	Access flag fault, level 1
001010	Access flag fault, level 2
001011	Access flag fault, level 3
001101	Permission fault, level 1
001110	Permission fault, level 2
001111	Permission fault, level 3
010000	Synchronous external abort, not on translation table walk
010101	Synchronous external abort, on translation table walk, level 1
010110	Synchronous external abort, on translation table walk, level 2
010111	Synchronous external abort, on translation table walk, level 3
011000	Synchronous parity or ECC error on memory access, not on translation table walk
011101	Synchronous parity or ECC error on memory access on translation table walk, level 1
011110	Synchronous parity or ECC error on memory access on translation table walk, level 2
011111	Synchronous parity or ECC error on memory access on translation table walk, level 3
100001	PC alignment fault
100010	Debug exception
110000	TLB conflict abort

All other values are reserved.

When the RAS Extension is implemented, 011000, 011101, 011110, and 011111, are reserved.

The lookup level associated with a fault is:

- For a fault generated on a translation table walk, the lookup level of the walk being performed.
- For a Translation fault, the lookup level of the translation table that gave the fault. If a fault occurs because a stage of address translation is disabled, or because the input address is outside the range specified by the appropriate base address register or registers, the fault is reported as a fault at level 1.
- For an Access flag fault, the lookup level of the translation table that gave the fault.
- For a Permission fault, including a Permission fault caused by hierarchical permissions, the lookup level of the final level of translation table accessed for the translation. That is, the lookup level of the translation table that returned a Block or Page descriptor.

## Accessing the IFSR32\_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
IFSR32_EL2	11	100	0101	0000	001

## Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3

## IFSR32\_EL2, Instruction Fault Status Register (EL2)

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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ISR\_EL1, Interrupt Status Register

The ISR\_EL1 characteristics are:

## Purpose

Shows whether any IRQ, FIQ, or SError interrupt is pending. In an implementation that includes EL2, when the register is accessed from Non-secure EL1, a pending interrupt or external abort might be physical or virtual, and the architecture does not provide any mechanism that software executing at Non-secure EL1 can use to determine whether a pending interrupt or external abort is physical or virtual. For all other accesses, any indicated interrupt or external abort must be physical.

This register is part of the Exception and fault handling registers functional group.

## Configuration

AArch64 System register ISR\_EL1 is architecturally mapped to AArch32 System register [ISR](#).

## Attributes

ISR\_EL1 is a 32-bit register.

## Field descriptions

The ISR\_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	A	I	F	0	0	0	0	0	0

### Bits [31:9]

Reserved, RES0.

### A, bit [8]

SError interrupt pending bit:

A	Meaning
0	No pending SError.
1	An SError interrupt is pending.

### I, bit [7]

IRQ pending bit. Indicates whether an IRQ interrupt is pending:

I	Meaning
0	No pending IRQ.
1	An IRQ interrupt is pending.

### F, bit [6]

FIQ pending bit. Indicates whether an FIQ interrupt is pending.

F	Meaning
0	No pending FIQ.
1	An FIQ interrupt is pending.

**Bits [5:0]**

Reserved, RES0.

**Accessing the ISR\_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
ISR_EL1	11	000	1100	0001	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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# LORC\_EL1, LORegion Control (EL1)

The LORC\_EL1 characteristics are:

## Purpose

Enables and disables LORegions, and selects the current LORegion descriptor.

This register is part of the Virtual memory control registers functional group.

## Configuration

If no LORegion descriptors are supported by the PE, then this register is RES0.

RW fields in this register reset to architecturally UNKNOWN values.

This register is introduced in ARMv8.1.

## Attributes

LORC\_EL1 is a 64-bit register.

## Field descriptions

The LORC\_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		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	DS										0	EN
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

### Bits [63:10]

Reserved, RES0.

### DS, bits [9:2]

Descriptor Select. Selects the current LORegion descriptor accessed by [LORSA\\_EL1](#), [LOREA\\_EL1](#), and [LORN\\_EL1](#).

The number of LORegion descriptors in IMPLEMENTATION DEFINED. The maximum number of LORegion descriptors supported is 256. If the number is less than 256, then bits[63:M+2] are RES0, where M is Log<sub>2</sub>(Number of LORegion descriptors supported by the implementation).

If this field points to an LORegion descriptor that is not supported by an implementation, then the registers [LORN\\_EL1](#), [LOREA\\_EL1](#), and [LORSA\\_EL1](#) are RES0.

### Bit [1]

Reserved, RES0.

### EN, bit [0]

Enable. Indicates whether LORegions are enabled:

EN	Meaning
0	Disabled. Memory accesses do not match any LORegions.
1	Enabled. Memory accesses may match a LORegion.

This bit is permitted to be cached in a TLB.

## Accessing the LORC\_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
LORC_EL1	11	000	1010	0100	011

## 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	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.TLOR](#)==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.TLOR](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==1 :

- If [SCR\\_EL3.TLOR](#)==1, Non-secure accesses to this register from EL1 and EL2 are trapped to EL3.

# LOREA\_EL1, LORegion End Address (EL1)

The LOREA\_EL1 characteristics are:

## Purpose

Holds the physical address of the end of the LORegion described in the current LORegion descriptor selected by [LORC\\_EL1](#).DS.

This register is part of the Virtual memory control registers functional group.

## Configuration

This register is RES0 if any of the following apply:

- No LORegion descriptors are supported by the PE.
- [LORC\\_EL1](#).DS points to a LORegion that is not supported by the PE.

RW fields in this register reset to architecturally UNKNOWN values.

This register is introduced in ARMv8.1.

## Attributes

LOREA\_EL1 is a 64-bit register.

## Field descriptions

The LOREA\_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	EA[51:48]				EA[47:16]																	
EA[47:16]																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		

Any of the fields in this register are permitted to be cached in a TLB.

### Bits [63:52]

Reserved, RES0.

### EA[51:48], bits [51:48]

In ARMv8.2:

Extension to EA[47:16]. See EA[47:16] for more details.

In ARMv8.1:

Reserved, RES0.

### EA[47:16], bits [47:16]

Bits [47:16] of the end physical address of an LORegion described in the current LORegion descriptor selected by [LORC\\_EL1](#).DS. Bits[15:0] of this address are defined to be 0xFFFF.

When ARMv8.2-LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, EA[51:48] form the upper part of the address value. Otherwise, for implementations with fewer than 52 physical address bits, EA[51:48] are RES0.

**Bits [15:0]**

Reserved, RES0.

**Accessing the LOREA\_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
LOREA_EL1	11	000	1010	0100	001

**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	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.TLOR](#)==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.TLOR](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==1 :

- If [SCR\\_EL3.TLOR](#)==1, Non-secure accesses to this register from EL1 and EL2 are trapped to EL3.

# LORID\_EL1, LORegionID (EL1)

The LORID\_EL1 characteristics are:

## Purpose

Indicates the number of LORegions and LORegion descriptors supported by the PE.

This register is part of the Virtual memory control registers functional group.

## Configuration

If no LORegion descriptors are implemented, then the registers [LORC\\_EL1](#), [LORN\\_EL1](#), [LOREA\\_EL1](#), and [LORSA\\_EL1](#) are RES0.

RW fields in this register reset to architecturally UNKNOWN values.

This register is introduced in ARMv8.1.

## Attributes

LORID\_EL1 is a 64-bit register.

## Field descriptions

The LORID\_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
0	0	0	0	0	0	0	0	LD								0	0	0	0	0	0	0	LR								
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:24]

Reserved, RES0.

### LD, bits [23:16]

Number of LORegion descriptors supported by the PE. This is an 8-bit binary number.

### Bits [15:8]

Reserved, RES0.

### LR, bits [7:0]

Number of LORegions supported by the PE. This is an 8-bit binary number.

---

#### Note

If LORID\_EL1 indicates that no LORegions are implemented, then LoadLOAcquire and StoreLORelease will behave as LoadAcquire and StoreRelease.

---

## Accessing the LORID\_EL1

This register can be read using MRS with the following syntax:

MRS &lt;Xt&gt;, &lt;systemreg&gt;

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
LORID_EL1	11	000	1010	0100	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.TLOR](#)==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.TLOR](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [SCR\\_EL3.TLOR](#)==1, accesses to this register from EL1 and EL2 are trapped to EL3.

# LORN\_EL1, LORegion Number (EL1)

The LORN\_EL1 characteristics are:

## Purpose

Holds the number of the LORegion described in the current LORegion descriptor selected by [LORC\\_EL1](#).DS.

This register is part of the Virtual memory control registers functional group.

## Configuration

This register is RES0 if any of the following apply:

- No LORegion descriptors are supported by the PE.
- [LORC\\_EL1](#).DS points to a LORegion that is not supported by the PE.

RW fields in this register reset to architecturally UNKNOWN values.

This register is introduced in ARMv8.1.

## Attributes

LORN\_EL1 is a 64-bit register.

## Field descriptions

The LORN\_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					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Num												
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					

Any of the fields in this register are permitted to be cached in a TLB.

### Bits [63:8]

Reserved, RES0.

### Num, bits [7:0]

Number of the LORegion described in the current LORegion descriptor selected by [LORC\\_EL1](#).DS.

The maximum number of LORegions supported by the PE is 256. If the maximum number is less than 256, then bits[8:N] are RES0, where N is (Log<sub>2</sub>(Number of LORegions supported by the PE)).

If this field points to a LORegion that is not supported by the PE, then the current LORegion descriptor does not match any LORegion.

## Accessing the LORN\_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
LORN_EL1	11	000	1010	0100	010

## 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	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.TLOR](#)==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.TLOR](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==1 :

- If [SCR\\_EL3.TLOR](#)==1, Non-secure accesses to this register from EL1 and EL2 are trapped to EL3.



# LORSA\_EL1, LORegion Start Address (EL1)

The LORSA\_EL1 characteristics are:

## Purpose

Indicates whether the current LORegion descriptor selected by [LORC\\_EL1](#).DS is enabled, and holds the physical address of the start of the LORegion.

This register is part of the Virtual memory control registers functional group.

## Configuration

This register is RES0 if any of the following apply:

- No LORegion descriptors are supported by the PE.
- [LORC\\_EL1](#).DS points to a LORegion that is not supported by the PE.

RW fields in this register reset to architecturally UNKNOWN values.

This register is introduced in ARMv8.1.

## Attributes

LORSA\_EL1 is a 64-bit register.

## Field descriptions

The LORSA\_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	SA[51:48]				SA[47:16]																
SA[47:16]																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Valid
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Any of the fields in this register are permitted to be cached in a TLB.

### Bits [63:52]

Reserved, RES0.

### SA[51:48], bits [51:48]

In ARMv8.2:

Extension to SA[47:16]. See SA[47:16] for more details.

In ARMv8.1:

Reserved, RES0.

### SA[47:16], bits [47:16]

Bits [47:16] of the start physical address of the LORegion described in the current LORegion descriptor selected by [LORC\\_EL1](#).DS. Bits[15:0] of this address are defined to be 0x0000.

When ARMv8.2-LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, SA[51:48] form the upper part of the address value. Otherwise, for implementations with fewer than 52 physical address bits, SA[51:48] are RES0.

**Bits [15:1]**

Reserved, RES0.

**Valid, bit [0]**

Indicates whether the current LORegion Descriptor is enabled.

Valid	Meaning
0	Disabled
1	Enabled

**Accessing the LORSA\_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
LORSA_EL1	11	000	1010	0100	000

**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	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.TLOR](#)==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.TLOR](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 and SCR\_EL3.NS==1 :

- If [SCR\\_EL3.TLOR](#)==1, Non-secure accesses to this register from EL1 and EL2 are trapped to EL3.

# MAIR\_EL1, Memory Attribute Indirection Register (EL1)

The MAIR\_EL1 characteristics are:

## Purpose

Provides the memory attribute encodings corresponding to the possible AttrIdx values in a Long-descriptor format translation table entry for stage 1 translations at EL1.

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch64 System register MAIR\_EL1 bits [31:0] are architecturally mapped to AArch32 System register [PRRR](#) when TTBCR.EAE==0.

AArch64 System register MAIR\_EL1 bits [31:0] are architecturally mapped to AArch32 System register [MAIRO](#) when TTBCR.EAE==1.

AArch64 System register MAIR\_EL1 bits [63:32] are architecturally mapped to AArch32 System register [NMRR](#) when TTBCR.EAE==0.

AArch64 System register MAIR\_EL1 bits [63:32] are architecturally mapped to AArch32 System register [MAIR1](#) when TTBCR.EAE==1.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

MAIR\_EL1 is a 64-bit register.

## Field descriptions

The MAIR\_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
<a href="#">Attr7</a>								<a href="#">Attr6</a>								<a href="#">Attr5</a>								<a href="#">Attr4</a>							
<a href="#">Attr3</a>								<a href="#">Attr2</a>								<a href="#">Attr1</a>								<a href="#">Attr0</a>							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

MAIR\_EL1 is permitted to be cached in a TLB.

### Attr<n>, bits [8n+7:8n], for n = 0 to 7

The memory attribute encoding for an AttrIdx[2:0] entry in a Long descriptor format translation table entry, where AttrIdx[2:0] gives the value of <n> in Attr<n>.

Bits [7:4] are encoded as follows:

Attr<n>[7:4]	Meaning
0000	Device memory. See encoding of Attr<n>[3:0] for the type of Device memory.
00RW, RW not 00	Normal memory, Outer Write-Through Transient
0100	Normal memory, Outer Non-cacheable
01RW, RW not 00	Normal memory, Outer Write-Back Transient
10RW	Normal memory, Outer Write-Through Non-transient
11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

The meaning of bits [3:0] depends on the value of bits [7:4]:

Attr<n>[3:0]	Meaning when Attr<n>[7:4] is 0000	Meaning when Attr<n>[7:4] is not 0000
0000	Device-nGnRnE memory	UNPREDICTABLE
00RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Transient
0100	Device-nGnRE memory	Normal memory, Inner Non-cacheable
01RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Transient
1000	Device-nGRE memory	Normal memory, Inner Write-Through Non- transient (RW=00)
10RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Non- transient
1100	Device-GRE memory	Normal memory, Inner Write-Back Non- transient (RW=00)
11RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Non- transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in some Attr<n> fields have the following meanings:

R or W	Meaning
0	No Allocate
1	Allocate

## Accessing the MAIR\_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
MAIR_EL1	11	000	1010	0010	000
MAIR_EL12	11	101	1010	0010	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
MAIR_EL1	x	x	0	-	RW	n/a	RW
MAIR_EL1	0	0	1	-	RW	RW	RW
MAIR_EL1	0	1	1	-	n/a	RW	RW
MAIR_EL1	1	0	1	-	RW	<a href="#">MAIR_EL2</a>	RW
MAIR_EL1	1	1	1	-	n/a	<a href="#">MAIR_EL2</a>	RW
MAIR_EL12	x	x	0	-	-	n/a	-
MAIR_EL12	0	0	1	-	-	-	-
MAIR_EL12	0	1	1	-	n/a	-	-
MAIR_EL12	1	0	1	-	-	RW	RW
MAIR_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic MAIR\_EL1 or MAIR\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

## 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.

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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# MAIR\_EL2, Memory Attribute Indirection Register (EL2)

The MAIR\_EL2 characteristics are:

## Purpose

Provides the memory attribute encodings corresponding to the possible AttrIdx values in a Long-descriptor format translation table entry for stage 1 translations at EL2.

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

## Configuration

AArch64 System register MAIR\_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HMAIR0](#).

AArch64 System register MAIR\_EL2 bits [63:32] are architecturally mapped to AArch32 System register [HMAIR1](#).

If EL2 is not implemented, this register is RES0 from EL3.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

MAIR\_EL2 is a 64-bit register.

## Field descriptions

The MAIR\_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
Attr7								Attr6								Attr5								Attr4							
Attr3								Attr2								Attr1								Attr0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

MAIR\_EL2 is permitted to be cached in a TLB.

### Attr<n>, bits [8n+7:8n], for n = 0 to 7

The memory attribute encoding for an AttrIdx[2:0] entry in a Long descriptor format translation table entry, where AttrIdx[2:0] gives the value of <n> in Attr<n>.

Bits [7:4] are encoded as follows:

Attr<n>[7:4]	Meaning
0000	Device memory. See encoding of Attr<n>[3:0] for the type of Device memory.
00RW, RW not 00	Normal memory, Outer Write-Through Transient
0100	Normal memory, Outer Non-cacheable
01RW, RW not 00	Normal memory, Outer Write-Back Transient
10RW	Normal memory, Outer Write-Through Non-transient
11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

The meaning of bits [3:0] depends on the value of bits [7:4]:

Attr<n>[3:0]	Meaning when Attr<n>[7:4] is 0000	Meaning when Attr<n>[7:4] is not 0000
0000	Device-nGnRnE memory	UNPREDICTABLE
00RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Transient
0100	Device-nGnRE memory	Normal memory, Inner Non-cacheable
01RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Transient
1000	Device-nGRE memory	Normal memory, Inner Write-Through Non- transient (RW=00)
10RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Non- transient
1100	Device-GRE memory	Normal memory, Inner Write-Back Non- transient (RW=00)
11RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Non- transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in some Attr<n> fields have the following meanings:

R or W	Meaning
0	No Allocate
1	Allocate

## Accessing the MAIR\_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
MAIR_EL2	11	100	1010	0010	000
MAIR_EL1	11	000	1010	0010	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
MAIR_EL2	x	x	0	-	-	n/a	RW
MAIR_EL2	0	0	1	-	-	RW	RW
MAIR_EL2	0	1	1	-	n/a	RW	RW
MAIR_EL2	1	0	1	-	-	RW	RW
MAIR_EL2	1	1	1	-	n/a	RW	RW
MAIR_EL1	x	x	0	-	<a href="#">MAIR_EL1</a>	n/a	<a href="#">MAIR_EL1</a>
MAIR_EL1	0	0	1	-	<a href="#">MAIR_EL1</a>	<a href="#">MAIR_EL1</a>	<a href="#">MAIR_EL1</a>
MAIR_EL1	0	1	1	-	n/a	<a href="#">MAIR_EL1</a>	<a href="#">MAIR_EL1</a>
MAIR_EL1	1	0	1	-	<a href="#">MAIR_EL1</a>	RW	<a href="#">MAIR_EL1</a>
MAIR_EL1	1	1	1	-	n/a	RW	<a href="#">MAIR_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic MAIR\_EL2 or MAIR\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.





# MAIR\_EL3, Memory Attribute Indirection Register (EL3)

The MAIR\_EL3 characteristics are:

## Purpose

Provides the memory attribute encodings corresponding to the possible AttrIdx values in a Long-descriptor format translation table entry for stage 1 translations at EL3.

This register is part of the Virtual memory control registers functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

MAIR\_EL3 is a 64-bit register.

## Field descriptions

The MAIR\_EL3 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
Attr7								Attr6								Attr5								Attr4							
Attr3								Attr2								Attr1								Attr0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

MAIR\_EL3 is permitted to be cached in a TLB.

### Attr<n>, bits [8n+7:8n], for n = 0 to 7

The memory attribute encoding for an AttrIdx[2:0] entry in a Long descriptor format translation table entry, where AttrIdx[2:0] gives the value of <n> in Attr<n>.

Bits [7:4] are encoded as follows:

Attr<n>[7:4]	Meaning
0000	Device memory. See encoding of Attr<n>[3:0] for the type of Device memory.
00RW, RW not 00	Normal memory, Outer Write-Through Transient
0100	Normal memory, Outer Non-cacheable
01RW, RW not 00	Normal memory, Outer Write-Back Transient
10RW	Normal memory, Outer Write-Through Non-transient
11RW	Normal memory, Outer Write-Back Non-transient

R = Outer Read-Allocate policy, W = Outer Write-Allocate policy.

The meaning of bits [3:0] depends on the value of bits [7:4]:

Attr<n>[3:0]	Meaning when Attr<n>[7:4] is 0000	Meaning when Attr<n>[7:4] is not 0000
0000	Device-nGnRnE memory	UNPREDICTABLE
00RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Transient
0100	Device-nGnRE memory	Normal memory, Inner Non-cacheable
01RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Transient
1000	Device-nGRE memory	Normal memory, Inner Write-Through Non- transient (RW=00)
10RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Through Non- transient
1100	Device-GRE memory	Normal memory, Inner Write-Back Non- transient (RW=00)
11RW, RW not 00	UNPREDICTABLE	Normal memory, Inner Write-Back Non- transient

R = Inner Read-Allocate policy, W = Inner Write-Allocate policy.

The R and W bits in some Attr<n> fields have the following meanings:

R or W	Meaning
0	No Allocate
1	Allocate

## Accessing the MAIR\_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
MAIR_EL3	11	110	1010	0010	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.

# MDCCINT\_EL1, Monitor DCC Interrupt Enable Register

The MDCCINT\_EL1 characteristics are:

## Purpose

Enables interrupt requests to be signaled based on the DCC status flags.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register MDCCINT\_EL1 is architecturally mapped to AArch32 System register [DBGDCCINT](#).

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 AArch64. Otherwise, on a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

MDCCINT\_EL1 is a 32-bit register.

## Field descriptions

The MDCCINT\_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	RX	TX	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

### Bit [31]

Reserved, RES0.

### RX, bit [30]

DCC interrupt request enable control for DTRRX. Enables a common COMMIRQ interrupt request to be signaled based on the DCC status flags.

RX	Meaning
0	No interrupt request generated by DTRRX.
1	Interrupt request will be generated on RXfull == 1.

If legacy COMMRX and COMMTX signals are implemented, then these are not affected by the value of this bit.

When this register has an architecturally-defined reset value, this field resets to 0.

### TX, bit [29]

DCC interrupt request enable control for DTRTX. Enables a common COMMIRQ interrupt request to be signaled based on the DCC status flags.

TX	Meaning
0	No interrupt request generated by DTRTX.
1	Interrupt request will be generated on TXfull == 0.

If legacy COMMRX and COMMTX signals are implemented, then these are not affected by the value of this bit.

When this register has an architecturally-defined reset value, this field resets to 0.

**Bits [28:0]**

Reserved, RES0.

**Accessing the MDCCINT\_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
MDCCINT_EL1	10	000	0000	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
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<sup>®</sup> 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 :

- 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.

# MDCCSR\_EL0, Monitor DCC Status Register

The MDCCSR\_EL0 characteristics are:

## Purpose

Main control register for the debug implementation, containing flow-control flags for the DCC. This is an internal, read-only view.

This register is part of the Debug registers functional group.

## Configuration

There are no configuration notes.

## Attributes

MDCCSR\_EL0 is a 32-bit register.

## Field descriptions

The MDCCSR\_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	RXfull	TXfull	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

### Bit [31]

Reserved, RES0.

### RXfull, bit [30]

DTRRX full. Read-only view of the equivalent bit in the [EDSCR](#).

### TXfull, bit [29]

DTRTX full. Read-only view of the equivalent bit in the [EDSCR](#).

### Bits [28:19]

Reserved, RES0.

### Bits [18:15]

RAZ/WI. Hardware must implement this field as RAZ/WI. Software must not rely on the register reading as zero, and must use a read-modify-write sequence to write to the register.

### Bits [14:13]

Reserved, RES0.

### Bit [12]

RAZ/WI. Hardware must implement this field as RAZ/WI. Software must not rely on the register reading as zero, and must use a read-modify-write sequence to write to the register.

**Bits [11:6]**

Reserved, RES0.

**Bits [5:2]**

RAZ/WI. Hardware must implement this field as RAZ/WI. Software must not rely on the register reading as zero, and must use a read-modify-write sequence to write to the register.

**Bits [1:0]**

Reserved, RES0.

## Accessing the MDCCSR\_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
MDCCSR_EL0	10	011	0000	0001	000

## 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<sup>®</sup> 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 [MDSCR\\_EL1.TDCC](#)==1, 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.TDA](#)==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.TDA](#)==1, read accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

# MDCR\_EL2, Monitor Debug Configuration Register (EL2)

The MDCR\_EL2 characteristics are:

## Purpose

Provides EL2 configuration options for self-hosted debug and the Performance Monitors Extension.

This register is part of:

- The Debug registers functional group.
- The Virtualization registers functional group.

## Configuration

AArch64 System register MDCR\_EL2 is architecturally mapped to AArch32 System register [HDCR](#).

If EL2 is not implemented, this register is RES0 from EL3.

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 AArch64. Otherwise, on a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

MDCR\_EL2 is a 32-bit register.

## Field descriptions

The MDCR\_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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	HPMD	0	0	TPMS	E2PB	TDRA	TDOSA	TDATDE	HPME	TPM	TPMCR					HPMN		

### Bits [31:18]

Reserved, RES0.

### HPMD, bit [17]

In ARMv8.2 and ARMv8.1:

Guest Performance Monitors Disable. This control prohibits event counting at EL2. Permitted values are:

HPMD	Meaning
0	Event counting allowed at EL2.
1	Event counting prohibited at EL2. In an ARMv8.1 implementation, event counting is prohibited unless enabled by the IMPLEMENTATION DEFINED authentication interface ExternalSecureNoninvasiveDebugEnabled().

This control applies only to:

- The event counters in the range [0..(HPMN-1)].
- If [PMCR\\_EL0](#).DP is set to 1, [PMCCNTR\\_EL0](#).

The other event counters are unaffected, and when [PMCR\\_EL0](#).DP is set to 0, [PMCCNTR\\_EL0](#) is unaffected.

When this register has an architecturally-defined reset value, this field resets to 0.

**In ARMv8.0:**

Reserved, RES0.

**Bits [16:15]**

Reserved, RES0.

**TPMS, bit [14]****In ARMv8.2:**

Trap Performance Monitor Sampling. When the Statistical Profiling Extension is implemented this field controls access to Statistical Profiling control registers from Non-secure EL1 and EL0. The possible values of this bit are:

TPMS	Meaning
0	Do not trap Statistical Profiling controls to EL2.
1	Accesses to Statistical Profiling controls at Non-secure EL1 generate a Trap exception to EL2.

If EL2 is not implemented, the PE behaves as if TPMS == 0, other than for a direct read of the register.

When the Statistical Profiling Extension is not implemented this field is reserved, RES0.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**E2PB, bits [13:12]****In ARMv8.2:**

EL2 Profiling Buffer. When the Statistical Profiling Extension is implemented this field controls the owning translation regime and access to Profiling Buffer control registers from Non-secure EL1. The possible values of this field are:

E2PB	Meaning
00	Profiling Buffer uses the EL2 stage 1 translation regime. Accesses to Profiling Buffer controls at Non-secure EL1 generate a Trap exception to EL2.
10	Profiling Buffer uses the EL1&0 stage 1 translation regime. Accesses to Profiling Buffer controls at Non-secure EL1 generate a Trap exception to EL2.
11	Profiling Buffer uses the EL1&0 stage 1 translation regime. Accesses to Profiling Buffer controls at Non-secure EL1 are not trapped to EL2.

All other values are reserved. If this field is programmed with a reserved value, the PE behaves as if this field has a defined value, other than for a direct read of the register. Software must not rely on the behavior of reserved values, as they might change in a future version of the architecture.

If EL2 is not implemented, the PE behaves as if E2PB == 0b11, other than for a direct read of the register.

When the Statistical Profiling Extension is not implemented this field is reserved, RES0.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**TDRA, bit [11]**

Trap Debug ROM Address register access. Traps Non-secure System register accesses to the Debug ROM registers to EL2. This trap is from:

- Non-secure EL0 using AArch32.
- Non-secure EL1, regardless of which Execution state it is using.



TDRA	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL0 and EL1 System register accesses to the Debug ROM registers are trapped to EL2, unless it is trapped by <a href="#">DBGDSCRExt</a> .UDCCdis or <a href="#">MDSCR_EL1</a> .TDCC.

The registers for which accesses are trapped are as follows:

AArch64: [MDRAR\\_EL1](#).

AArch32: [DBGDRAR](#), [DBGDSAR](#).

If [MDCR\\_EL2](#).TDE == 1 or [HCR\\_EL2](#).TGE == 1, behavior is as if this bit is 1 other than for the purpose of a direct read.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## TDOSA, bit [10]

Trap debug OS-related register access. Traps Non-secure EL1 System register accesses to the powerdown debug registers to EL2, from both Execution states:

TDOSA	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL1 System register accesses to the powerdown debug registers are trapped to EL2.

The registers for which accesses are trapped are as follows:

AArch64: [OSLAR\\_EL1](#), [OSLSR\\_EL1](#), [OSDLR\\_EL1](#), and the [DBGPRCR\\_EL1](#).

AArch32: [DBGOSLSR](#), [DBGOSLAR](#), [DBGOSDLR](#), and the [DBGPRCR](#).

AArch64 and AArch32: Any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit.

### Note

These registers are not accessible at EL0.

If [MDCR\\_EL2](#).TDE == 1 or [HCR\\_EL2](#).TGE == 1, behavior is as if this bit is 1 other than for the purpose of a direct read.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## TDA, bit [9]

Trap Debug Access. Traps Non-secure EL0 and EL1 System register accesses to those debug System registers that are not trapped by either of the following:

- MDCR\_EL2.TDRA.
- MDCR\_EL2.TDOSA.

TDA	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL0 or EL1 System register accesses to the debug registers, other than the registers trapped by MDCR_EL2.TDRA and MDCR_EL2.TDOSA, are trapped to EL2, from both Execution states, unless it is trapped by <a href="#">DBGDSCRExt</a> .UDCCdis or <a href="#">MDSCR_EL1</a> .TDCC.

Traps of AArch32 accesses to [DBGDTRRXint](#) and [DBGDTRTXint](#) are ignored in Debug state.

Traps of AArch64 accesses to [DBGDTR\\_EL0](#), [DBGDTRRX\\_EL0](#), and [DBGDTRTX\\_EL0](#) are ignored in Debug state.

If MDCR\_EL2.TDE == 1 or [HCR\\_EL2](#).TGE == 1, behavior is as if this bit is 1 other than for the purpose of a direct read.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**TDE, bit [8]**

Trap Debug exceptions. The possible values of this field are:

TDE	Meaning
0	This control has no effect on the routing of debug exceptions, and has no effect on Non-secure accesses to debug registers.
1	In Non-secure state: <ul style="list-style-type: none"> <li>• Debug exceptions generated at EL1 or EL0 are routed to EL2.</li> <li>• The MDCR_EL2.{TDRA, TDOSA, TDA} fields are treated as being 1 for all purposes other than returning the result of a direct read of the register.</li> </ul>

When [HCR\\_EL2.TGE](#) == 1, the PE behaves as if the value of this field is 1 for all purposes other than returning the value of a direct read of the register.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**HPME, bit [7]**

Hypervisor Performance Monitors Counters Enable. The possible values of this bit are:

HPME	Meaning
0	EL2 Performance Monitors counters disabled.
1	EL2 Performance Monitors counters enabled.

When the value of this bit is 1, the Performance Monitors counters that are reserved for use from EL2 or Secure state are enabled. For more information see the description of the HPMN field.

If the Performance Monitors Extension is not implemented, this field is RES0.

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.

**TPM, bit [6]**

Trap Performance Monitors accesses. Traps Non-secure EL0 and EL1 accesses to all Performance Monitors registers to EL2, from both Execution states:

TPM	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL0 and EL1 accesses to all Performance Monitors registers are trapped to EL2.

**Note**

EL2 does not provide traps on Performance Monitor register accesses through the optional memory-mapped external debug interface.

If the Performance Monitors Extension is not implemented, this field is RES0.

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.

**TPMCR, bit [5]**

Trap [PMCR\\_EL0](#) or [PMCR](#) accesses. Traps Non-secure EL0 and EL1 accesses to the [PMCR\\_EL0](#) or [PMCR](#) to EL2.

TPMCR	Meaning
0	This control does not cause any instructions to be trapped.
1	Non-secure EL0 and EL1 accesses to the <a href="#">PMCR_EL0</a> or <a href="#">PMCR</a> are trapped to EL2, unless it is trapped by <a href="#">PMUSERENR.EN</a> or <a href="#">PMUSERENR_EL0.EN</a> .

**Note**

EL2 does not provide traps on Performance Monitor register accesses through the optional memory-mapped external debug interface.

If the Performance Monitors Extension is not implemented, this field is RES0.

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.

### HPMN, bits [4:0]

Defines the number of Performance Monitors counters that are accessible from Non-secure EL0 and EL1 modes.

If the Performance Monitors Extension is not implemented, this field is RES0.

In Non-secure state, HPMN divides the Performance Monitors counters as follows. For counter  $n$  in Non-secure state:

- If  $n$  is in the range  $0 \leq n < \text{HPMN}$ , the counter is accessible from EL1 and EL2, and from EL0 if permitted by [PMUSERENR\\_EL0.PMCR\\_EL0.E](#) enables the operation of counters in this range.
- If  $n$  is in the range  $\text{HPMN} \leq n < \text{PMCR\_EL0.N}$ , the counter is accessible only from EL2 and from Secure state. MDCR\_EL2.HPME enables the operation of counters in this range.

If this field is set to 0, or to a value larger than [PMCR\\_EL0.N](#), then the following CONSTRAINED UNPREDICTABLE behavior applies:

- The value returned by a direct read of MDCR\_EL2.HPMN is UNKNOWN.
- Either:
  - An UNKNOWN number of counters are reserved for EL2 use. That is, the PE behaves as if MDCR\_EL2.HPMN is set to an UNKNOWN non-zero value less than [PMCR\\_EL0.N](#).
  - All counters are reserved for EL2 use, meaning no counters are accessible from Non-secure EL1 and Non-secure EL0.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to the value of [PMCR\\_EL0.N](#).

## Accessing the MDCR\_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
MDCR_EL2	11	100	0001	0001	001

## 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 EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDA==1, accesses to this register from EL2 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# MDCR\_EL3, Monitor Debug Configuration Register (EL3)

The MDCR\_EL3 characteristics are:

## Purpose

Provides EL3 configuration options for self-hosted debug and the Performance Monitors Extension.

This register is part of:

- The Debug registers functional group.
- The Security registers functional group.

## Configuration

AArch64 System register MDCR\_EL3 can be mapped to AArch32 System register [SDCR](#), but this is not architecturally mandated.

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 AArch64. Otherwise, on a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

MDCR\_EL3 is a 32-bit register.

## Field descriptions

The MDCR\_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
0	0	0	0	0	0	0	0	0	0	<a href="#">EPMAD</a>	<a href="#">EDAD</a>	0	0	<a href="#">SPME</a>	<a href="#">SDD</a>	<a href="#">SPD32</a>	<a href="#">NSPB</a>	0	<a href="#">TDOSA</a>	<a href="#">TDA</a>	0	0	<a href="#">TPM</a>	0	0	0	0	0	0	0	0

### Bits [31:22]

Reserved, RES0.

### EPMAD, bit [21]

External debug interface Performance Monitors registers disable. This disables access to these registers by an external debugger:

EPMAD	Meaning
0	Access to Performance Monitors registers from external debugger is permitted.
1	Access to Performance Monitors registers from external debugger is disabled, unless overridden by the IMPLEMENTATION DEFINED authentication interface.

If the Performance Monitors Extension is not implemented or does not support external debug interface accesses 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.

### EDAD, bit [20]

External debug interface breakpoint and watchpoint register access disable. This disables access to these registers by an external debugger:

EDAD	Meaning
0	Access to breakpoint and watchpoint registers from external debugger is permitted.
1	Access to breakpoint and watchpoint registers from external debugger is disabled, unless overridden by the IMPLEMENTATION DEFINED authentication interface.

When this register has an architecturally-defined reset value, this field resets to 0.

### Bits [19:18]

Reserved, RES0.

### SPME, bit [17]

Secure Performance Monitors enable. This allows event counting in Secure state:

SPME	Meaning
0	Event counting prohibited in Secure state. In an ARMv8.0 or ARMv8.1 implementation, event counting is prohibited unless ExternalSecureNoninvasiveDebugEnabled() is TRUE, meaning this control is overridden by the IMPLEMENTATION DEFINED authentication interface.
1	Event counting allowed in Secure state.

If the Performance Monitors Extension is not implemented, this field is RES0.

When this register has an architecturally-defined reset value, this field resets to 0.

### SDD, bit [16]

AArch64 Secure self-hosted invasive debug disable. Disables Software debug exceptions in Secure state, other than Breakpoint Instruction exceptions.

SDD	Meaning
0	Debug exceptions from Secure EL0 are enabled, and debug exceptions from Secure EL1 are enabled if the value of <a href="#">MDSCR_EL1.KDE</a> is 1 and the value of PSTATE.D is 0.
1	Debug exceptions, other than Breakpoint Instruction exceptions, are disabled from all Exception levels in Secure state.

The SDD bit is ignored unless both of the following are true:

- The PE is in Secure state.
- Secure EL1 is using AArch64.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### SPD32, bits [15:14]

AArch32 Secure self-hosted privileged invasive debug control. Enables or disables debug exceptions from Secure EL1 using AArch32, other than Breakpoint Instruction exceptions. Valid values for this field are:

SPD32	Meaning
00	Legacy mode. Debug exceptions from Secure EL1 are enabled by the IMPLEMENTATION DEFINED authentication interface.
10	Secure privileged debug disabled. Debug exceptions from Secure EL1 are disabled.
11	Secure privileged debug enabled. Debug exceptions from Secure EL1 are enabled.

Other values are reserved, and have the CONSTRAINED UNPREDICTABLE behavior that they must have the same behavior as 0b00. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

This field has no effect on Breakpoint Instruction exceptions. These are always enabled.

This field is:

- Ignored if either the PE is in Non-secure state or Secure EL1 is using AArch64.
- RES0 if the implementation does not support EL1 using AArch32.

If Secure EL1 is using AArch32 then:

- If debug exceptions from Secure EL1 are enabled, then debug exceptions from Secure EL0 are also enabled.
- Otherwise, debug exceptions from Secure EL0 are enabled only if the value of [SDER32\\_EL3.SUIDEN](#) is 1.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to a value that is architecturally UNKNOWN.

### NSPB, bits [13:12]

In ARMv8.2:

Non-secure Profiling Buffer. When the Statistical Profiling Extension is implemented, this field controls the owning translation regime and accesses to Statistical Profiling and Profiling Buffer control registers. The possible values of this field are:

NSPB	Meaning
00	Profiling Buffer uses Secure Virtual Addresses. Statistical Profiling enabled in Secure state and disabled in Non-secure state. Accesses to Statistical Profiling and Profiling Buffer controls at EL2 and EL1 in both security states generate Trap exceptions to EL3.
01	Profiling Buffer uses Secure Virtual Addresses. Statistical Profiling enabled in Secure state and disabled in Non-secure state. Accesses to Statistical Profiling and Profiling Buffer controls in Non-secure state generate Trap exceptions to EL3.
10	Profiling Buffer uses Non-secure Virtual Addresses. Statistical Profiling enabled in Non-secure state and disabled in Secure state. Accesses to Statistical Profiling and Profiling Buffer controls at EL2 and EL1 in both security states generate Trap exceptions to EL3.
11	Profiling Buffer uses Non-secure Virtual Addresses. Statistical Profiling enabled in Non-secure state and disabled in Secure state. Accesses to Statistical Profiling and Profiling Buffer controls at Secure EL1 generate Trap exceptions to EL3.

If EL3 is not implemented and the PE executes in Non-secure state, the PE behaves as if NSPB == 0b11.

If EL3 is not implemented and the PE executes in Secure state, the PE behaves as if NSPB == 0b01.

When the Statistical Profiling Extension is not implemented this field is reserved, RES0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

### Bit [11]

Reserved, RES0.

### TDOSA, bit [10]

Trap debug OS-related register access. Traps EL2 and EL1 System register accesses to the powerdown debug registers to EL3:

TDOSA	Meaning
0	This control does not cause any instructions to be trapped.
1	EL2 and EL1 System register accesses to the powerdown debug registers are trapped to EL3, unless it is trapped by <a href="#">HDCR.TDOSA</a> or <a href="#">MDCR_EL2.TDOSA</a> .

The registers for which accesses are trapped are as follows:

AArch64: [OSLAR\\_EL1](#), [OSLSR\\_EL1](#), [OSDLR\\_EL1](#), [DBGPRCR\\_EL1](#).

AArch32: [DBGOSLAR](#), [DBGOSLSR](#), [DBGOSDLR](#), [DBGPRCR](#).

AArch64 and AArch32: Any IMPLEMENTATION DEFINED register with similar functionality that the implementation specifies as trapped by this bit.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### TDA, bit [9]

Trap Debug Access. Traps EL2, EL1, and EL0 System register accesses to those debug System registers that cannot be trapped using the MDCR\_EL3.TDOSA field. When MDCR\_EL3.TDA is:

TDA	Meaning
0	This control does not cause any instructions to be trapped.
1	EL0, EL1, and EL2 accesses to the debug registers, other than the registers that can be trapped by MDCR_EL3.TDOSA, are trapped to EL3, from both Security states and both Execution states, unless it is trapped by <a href="#">DBGDSCRExt.UDCCdis</a> , <a href="#">MDCR_EL1.TDCC</a> , <a href="#">HDCR.TDA</a> or <a href="#">MDCR_EL2.TDA</a> .

Traps of AArch32 accesses to [DBGDTRRXint](#) and [DBGDTRTXint](#) are ignored in Debug state.

Traps of AArch64 accesses to [DBGDTR\\_EL0](#), [DBGDTRRX\\_EL0](#), and [DBGDTRTX\\_EL0](#) are ignored in Debug state.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Bits [8:7]

Reserved, RES0.

#### TPM, bit [6]

Trap Performance Monitors accesses. Traps EL2, EL1, and EL0 accesses to all Performance Monitors registers to EL3, from both Security states and both Execution states.

TPM	Meaning
0	This control does not cause any instructions to be trapped.
1	EL2, EL1, and EL0 System register accesses to all Performance Monitors registers are trapped to EL3, unless it is trapped by <a href="#">HDCR.TPM</a> or <a href="#">MDCR_EL2.TPM</a> .

If the Performance Monitors Extension is not implemented, this field is RES0.

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 [5:0]

Reserved, RES0.

## Accessing the MDCR\_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
MDCR_EL3	11	110	0001	0011	001

## 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.





# MDRAR\_EL1, Monitor Debug ROM Address Register

The MDRAR\_EL1 characteristics are:

## Purpose

Defines the base physical address of a 4KB-aligned memory-mapped debug component, usually a ROM table that locates and describes the memory-mapped debug components in the system. ARMv8 deprecates any use of this register.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register MDRAR\_EL1 is architecturally mapped to AArch32 System register [DBGDRAR](#).

## Attributes

MDRAR\_EL1 is a 64-bit register.

## Field descriptions

The MDRAR\_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	ROMADDR[51:48]				ROMADDR[47:12]																											
ROMADDR[47:12]												0												0				0				0				0				Valid			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0												

### Bits [63:52]

Reserved, RES0.

### ROMADDR[51:48], bits [51:48]

In ARMv8.2:

Extension to ROMADDR[47:12]. See ROMADDR[47:12] for more details.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

### ROMADDR[47:12], bits [47:12]

Bits[47:12] of the ROM table physical address.

When ARMv8.2-LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, ROMADDR[52:49] forms the upper part of the address value. Otherwise, ROMADDR[52:49] is RES0.

If the physical address size in bits (PAsize) is less than 52 then the register bits corresponding to ROMADDR [51:PAsize] are RES0.

Bits [11:0] of the ROM table physical address are zero.

ARM strongly recommends that bits ROMADDR[(PAsize-1):32] are zero in any system that supports AArch32 at the highest implemented Exception level.

In an implementation that includes EL3, ROMADDR is an address in Non-secure memory. It is IMPLEMENTATION DEFINED whether the ROM table is also accessible in Secure memory.

**Bits [11:2]**

Reserved, RES0.

**Valid, bits [1:0]**

This field indicates whether the ROM Table address is valid. The permitted values of this field are:

Valid	Meaning
00	ROM Table address is not valid. Software must ignore ROMADDR.
11	ROM Table address is valid.

Other values are reserved.

**Accessing the MDRAR\_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
MDRAR_EL1	10	000	0001	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.

**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 :

- If [MDCR\\_EL2.TDRA](#)==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3.TDA](#)==1, read accesses to this register from EL1 and EL2 are trapped to EL3.

# MDSCR\_EL1, Monitor Debug System Control Register

The MDSCR\_EL1 characteristics are:

## Purpose

Main control register for the debug implementation.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register MDSCR\_EL1 is architecturally mapped to AArch32 System register [DBGDSCRExt](#).

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 AArch64. Otherwise, on a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

MDSCR\_EL1 is a 32-bit register.

## Field descriptions

The MDSCR\_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	RXfull	TXfull	0	RXO	TXU	0	0	INTdis	TDA	0	SC2	0	0	0	MDE	HDE	KDE	TDCC	0	0	0	0	0	ERR	0	0	0	0	0	SS	

### Bit [31]

Reserved, RES0.

### RXfull, bit [30]

Used for save/restore of [EDSCR](#).RXfull.

When [OSLSR\\_EL1](#).OSLK == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [OSLSR\\_EL1](#).OSLK == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR](#).RXfull.

Reads and writes of this bit are indirect accesses to [EDSCR](#).RXfull.

The architected behavior of this field determines the value it returns after a reset.

### TXfull, bit [29]

Used for save/restore of [EDSCR](#).TXfull.

When [OSLSR\\_EL1](#).OSLK == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [OSLSR\\_EL1](#).OSLK == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR](#).TXfull.

Reads and writes of this bit are indirect accesses to [EDSCR](#).TXfull.

The architected behavior of this field determines the value it returns after a reset.

**Bit [28]**

Reserved, RES0.

**R XO, bit [27]**

Used for save/restore of [EDSCR.R XO](#).

When [OSLSR\\_EL1.OSLK](#) == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [OSLSR\\_EL1.OSLK](#) == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR.R XO](#).

Reads and writes of this bit are indirect accesses to [EDSCR.R XO](#).

The architected behavior of this field determines the value it returns after a reset.

**TXU, bit [26]**

Used for save/restore of [EDSCR.T XU](#).

When [OSLSR\\_EL1.OSLK](#) == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [OSLSR\\_EL1.OSLK](#) == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR.T XU](#).

Reads and writes of this bit are indirect accesses to [EDSCR.T XU](#).

The architected behavior of this field determines the value it returns after a reset.

**Bits [25:24]**

Reserved, RES0.

**INTdis, bits [23:22]**

Used for save/restore of [EDSCR.INTdis](#).

When [OSLSR\\_EL1.OSLK](#) == 0 (the OS lock is unlocked), this field is RO, and software must treat it as UNK/SBZP.

When [OSLSR\\_EL1.OSLK](#) == 1 (the OS lock is locked), this field is RW and holds the value of [EDSCR.INTdis](#).

Reads and writes of this field are indirect accesses to [EDSCR.INTdis](#).

The architected behavior of this field determines the value it returns after a reset.

**TDA, bit [21]**

Used for save/restore of [EDSCR.TDA](#).

When [OSLSR\\_EL1.OSLK](#) == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [OSLSR\\_EL1.OSLK](#) == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR.TDA](#).

Reads and writes of this bit are indirect accesses to [EDSCR.TDA](#).

The architected behavior of this field determines the value it returns after a reset.

**Bit [20]**

Reserved, RES0.

**Bit [19]**

In ARMv8.2 and ARMv8.0:

Reserved, RES0.

**In ARMv8.1:**

Used for save/restore of [EDSCR](#).SC2.

When [OSLSR\\_EL1](#).OSLK == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [OSLSR\\_EL1](#).OSLK == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR](#).SC2.

Reads and writes of this bit are indirect accesses to [EDSCR](#).SC2.

If the PC Sample-based Profiling Extension is not implemented, then this field is RES0.

**Bits [18:16]**

RAZ/WI. Hardware must implement this field as RAZ/WI. Software must not rely on the register reading as zero, and must use a read-modify-write sequence to write to the register.

**MDE, bit [15]**

Monitor debug events. Enable Breakpoint, Watchpoint, and Vector Catch exceptions.

MDE	Meaning
0	Breakpoint, Watchpoint, and Vector Catch exceptions disabled.
1	Breakpoint, Watchpoint, and Vector Catch exceptions enabled.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**HDE, bit [14]**

Used for save/restore of [EDSCR](#).HDE.

When [OSLSR\\_EL1](#).OSLK == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [OSLSR\\_EL1](#).OSLK == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR](#).HDE.

Reads and writes of this bit are indirect accesses to [EDSCR](#).HDE.

The architected behavior of this field determines the value it returns after a reset.

**KDE, bit [13]**

Local (kernel) debug enable. If EL<sub>D</sub> is using AArch64, enable debug exceptions within EL<sub>D</sub>. Permitted values are:

KDE	Meaning
0	Debug exceptions, other than Breakpoint Instruction exceptions, disabled within EL <sub>D</sub> .
1	All debug exceptions enabled within EL <sub>D</sub> .

RES0 if EL<sub>D</sub> is using AArch32.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**TDCC, bit [12]**

Traps EL0 accesses to the DCC registers to EL1, from both Execution states:

TDCC	Meaning
0	This control does not cause any instructions to be trapped.
1	EL0 using AArch64: EL0 accesses to the <a href="#">MDCCSR_EL0</a> , <a href="#">DBGDTR_EL0</a> , <a href="#">DBGDTRTX_EL0</a> , and <a href="#">DBGDTRRX_EL0</a> registers are trapped to EL1. EL0 using AArch32: EL0 accesses to the <a href="#">DBGDSCRint</a> , <a href="#">DBGDTRRXint</a> , <a href="#">DBGDTRTXint</a> , <a href="#">DBGDIDR</a> , <a href="#">DBGDSAR</a> , and <a href="#">DBGDRAR</a> registers are trapped to EL1.

**Note**

All accesses to these AArch32 registers are trapped, including LDC and STC accesses to [DBGDTRTXint](#) and [DBGDTRRXint](#), and MRRC accesses to [DBGDSAR](#) and [DBGDRAR](#).

Traps of AArch32 accesses to the [DBGDTRRXint](#) and [DBGDTRTXint](#) are ignored in Debug state.

Traps of AArch64 accesses to [DBGDTR\\_EL0](#), [DBGDTRRX\\_EL0](#), and [DBGDTRTX\\_EL0](#) are ignored in Debug state.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Bits [11:7]**

Reserved, RES0.

**ERR, bit [6]**

Used for save/restore of [EDSCR.ERR](#).

When [OSLSR\\_EL1.OSLK](#) == 0 (the OS lock is unlocked), this bit is RO, and software must treat it as UNK/SBZP.

When [OSLSR\\_EL1.OSLK](#) == 1 (the OS lock is locked), this bit is RW and holds the value of [EDSCR.ERR](#).

Reads and writes of this bit are indirect accesses to [EDSCR.ERR](#).

The architected behavior of this field determines the value it returns after a reset.

**Bits [5:1]**

Reserved, RES0.

**SS, bit [0]**

Software step control bit. If EL<sub>D</sub> is using AArch64, enable Software step. Permitted values are:

SS	Meaning
0	Software step disabled
1	Software step enabled.

RES0 if EL<sub>D</sub> is using AArch32.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the MDSCR\_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
MDSCR_EL1	10	000	0000	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
x	0	1	-	RW	RW	RW
x	1	1	-	n/a	RW	RW

This table applies to all instructions that can access this register.

Individual fields within this register might have restricted accessibility when [OSLSR\\_EL1.OSLK](#) == 0 (the OS lock is unlocked.) See the field descriptions for more detail.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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 :

- 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# 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 'Identification registers, functional group' in the ARMv8 ARM, section G4.18.1.

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.

# MPIDR\_EL1, Multiprocessor Affinity Register

The MPIDR\_EL1 characteristics are:

## Purpose

In a multiprocessor system, provides an additional PE identification mechanism for scheduling purposes.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register MPIDR\_EL1 is architecturally mapped to AArch32 System register [MPIDR](#).

The assigned value of the [MPIDR](#).{Aff2, Aff1, Aff0} or MPIDR\_EL1.{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole.

In a uniprocessor system ARM recommends that each Aff<n> field of this register returns a value of 0.

## Attributes

MPIDR\_EL1 is a 64-bit register.

## Field descriptions

The MPIDR\_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	Aff3									
1	U	0	0	0	0	0	MT	Aff2								Aff1								Aff0									
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

### Bits [63:40]

Reserved, RES0.

### Aff3, bits [39:32]

Affinity level 3. Highest level affinity field.

### Bit [31]

Reserved, RES1.

### U, bit [30]

Indicates a Uniprocessor system, as distinct from PE 0 in a multiprocessor system. The possible values of this bit are:

U	Meaning
0	Processor is part of a multiprocessor system.
1	Processor is part of a uniprocessor system.

### Bits [29:25]

Reserved, RES0.

**MT, bit [24]**

Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. The possible values of this bit are:

MT	Meaning
0	Performance of PEs at the lowest affinity level is largely independent.
1	Performance of PEs at the lowest affinity level is very interdependent.

**Aff2, bits [23:16]**

Affinity level 2. Second highest level affinity field.

**Aff1, bits [15:8]**

Affinity level 1. Third highest level affinity field.

**Aff0, bits [7:0]**

Affinity level 0. Lowest level affinity field.

## Accessing the MPIDR\_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
MPIDR_EL1	11	000	0000	0000	101

## 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.

# MVFR0\_EL1, AArch32 Media and VFP Feature Register 0

The MVFR0\_EL1 characteristics are:

## Purpose

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with [MVFR1\\_EL1](#) and [MVFR2\\_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 Floating-point registers functional group.
- The Identification registers functional group.

## Configuration

AArch64 System register MVFR0\_EL1 is architecturally mapped to AArch32 System register [MVFR0](#).

In an implementation where at least one Exception level supports execution in AArch32 state, but there is no support for Advanced SIMD and floating-point operation, this register is RAZ.

In an AArch64-only implementation, this register is UNKNOWN.

## Attributes

MVFR0\_EL1 is a 32-bit register.

## Field descriptions

The MVFR0\_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
FPRound				FPShVec				FPSqrt				FPDivide				FPTrap				FPDP				FPSP				SIMDReg			

### FPRound, bits [31:28]

Floating-Point Rounding modes. Indicates whether the floating-point implementation provides support for rounding modes. Defined values are:

FPRound	Meaning
0000	Not implemented, or only Round to Nearest mode supported, except that Round towards Zero mode is supported for VCVT instructions that always use that rounding mode regardless of the <a href="#">FPSCR</a> setting.
0001	All rounding modes supported.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

### FPShVec, bits [27:24]

Short Vectors. Indicates whether the floating-point implementation provides support for the use of short vectors. Defined values are:

FPShVec	Meaning
0000	Short vectors not supported.
0001	Short vector operation supported.

All other values are reserved.

In ARMv8-A the only permitted value is 0000.

### FPSqrt, bits [23:20]

Square Root. Indicates whether the floating-point implementation provides support for the ARMv6 VFP square root operations. Defined values are:

FPSqrt	Meaning
0000	Not supported in hardware.
0001	Supported.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

The VSQRT.F32 instruction also requires the single-precision floating-point attribute, bits [7:4], and the VSQRT.F64 instruction also requires the double-precision floating-point attribute, bits [11:8].

### FPDivide, bits [19:16]

Indicates whether the floating-point implementation provides support for VFP divide operations. Defined values are:

FPDivide	Meaning
0000	Not supported in hardware.
0001	Supported.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

The VDIV.F32 instruction also requires the single-precision floating-point attribute, bits [7:4], and the VDIV.F64 instruction also requires the double-precision floating-point attribute, bits [11:8].

### FPTrap, bits [15:12]

Floating Point Exception Trapping. Indicates whether the floating-point implementation provides support for exception trapping. Defined values are:

FPTrap	Meaning
0000	Not supported.
0001	Supported.

All other values are reserved.

A value of 0001 indicates that, when the corresponding trap is enabled, a floating-point exception generates an exception.

### FPDP, bits [11:8]

Double Precision. Indicates whether the floating-point implementation provides support for double-precision operations. Defined values are:

FPDP	Meaning
0000	Not supported in hardware.
0001	Supported, VFPv2.
0010	Supported, VFPv3, VFPv4, or ARMv8. VFPv3 and ARMv8 add an instruction to load a double-precision floating-point constant, and conversions between double-precision and fixed-point values.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0010.

A value of 0b0001 or 0b0010 indicates support for all VFP double-precision instructions in the supported version of VFP, except that, in addition to this field being nonzero:

- VSQRT.F64 is only available if the Square root field is 0001.
- VDIV.F64 is only available if the Divide field is 0001.
- Conversion between double-precision and single-precision is only available if the single-precision field is nonzero.

### FPSP, bits [7:4]

Single Precision. Indicates whether the floating-point implementation provides support for single-precision operations. Defined values are:

FPSP	Meaning
0000	Not supported in hardware.
0001	Supported, VFPv2.
0010	Supported, VFPv3 or VFPv4. VFPv3 adds an instruction to load a single-precision floating-point constant, and conversions between single-precision and fixed-point values.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0010.

A value of 0b0001 or 0b0010 indicates support for all VFP single-precision instructions in the supported version of VFP, except that, in addition to this field being nonzero:

- VSQRT.F32 is only available if the Square root field is 0001.
- VDIV.F32 is only available if the Divide field is 0001.
- Conversion between double-precision and single-precision is only available if the double-precision field is nonzero.

### SIMDReg, bits [3:0]

Advanced SIMD registers. Indicates whether the Advanced SIMD and floating-point implementation provides support for the Advanced SIMD and floating-point register bank. Defined values are:

SIMDReg	Meaning
0000	The implementation has no Advanced SIMD and floating-point support.
0001	The implementation includes floating-point support with 16 x 64-bit registers.
0010	The implementation includes Advanced SIMD and floating-point support with 32 x 64-bit registers.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0010.

## Accessing the MVFR0\_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
MVFR0_EL1	11	000	0000	0011	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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# MVFR1\_EL1, AArch32 Media and VFP Feature Register 1

The MVFR1\_EL1 characteristics are:

## Purpose

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with [MVFR0\\_EL1](#) and [MVFR2\\_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 Floating-point registers functional group.
- The Identification registers functional group.

## Configuration

AArch64 System register MVFR1\_EL1 is architecturally mapped to AArch32 System register [MVFR1](#).

In an implementation where at least one Exception level supports execution in AArch32 state, but there is no support for Advanced SIMD and floating-point operation, this register is RAZ.

In an AArch64-only implementation, this register is UNKNOWN.

## Attributes

MVFR1\_EL1 is a 32-bit register.

## Field descriptions

The MVFR1\_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
SIMDFMAC				FPHP				SIMDHP				SIMDSP				SIMDInt				SIMDLS				FPDNaN				FPFtZ			

### SIMDFMAC, bits [31:28]

Advanced SIMD Fused Multiply-Accumulate. Indicates whether the Advanced SIMD implementation provides fused multiply accumulate instructions. Defined values are:

SIMDFMAC	Meaning
0000	Not implemented.
0001	Implemented.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

The Advanced SIMD and floating-point implementations must provide the same level of support for these instructions.

### FPHP, bits [27:24]

Floating Point Half Precision. Indicates the level of half-precision floating-point support. Defined values are:

FPHP	Meaning
0000	Not supported.
0001	Floating-point half-precision conversion instructions are supported for conversion between single-precision and half-precision.
0010	Floating-point half-precision conversion instructions are supported for conversion between single-precision and half-precision and between double-precision and half-precision.
0011	As for 0010, and also includes support for half-precision floating-point arithmetic.

All other values are reserved.

The permitted values are:

- 0000 in an implementation without floating-point support.
- 0010 in an implementation with floating-point support, that does not include the ARMv8.2-FP16 extension.
- 0011 in an implementation with floating-point support, that includes the ARMv8.2-FP16 extension.

### SIMDHP, bits [23:20]

Advanced SIMD Half Precision. Indicates the level of half-precision floating-point support. Defined values are:

SIMDHP	Meaning
0000	Not supported.
0001	SIMD half-precision conversion instructions are supported for conversion between single-precision and half-precision.
0010	As for 0010, and also includes support for half-precision floating-point arithmetic.

All other values are reserved.

The permitted values are:

- 0000 in an implementation without SIMD floating-point support.
- 0001 in an implementation with SIMD floating-point support, that does not include the ARMv8.2-FP16 extension.
- 0010 in an implementation with SIMD floating-point support, that includes the ARMv8.2-FP16 extension.

### SIMDSP, bits [19:16]

Advanced SIMD Single Precision. Indicates whether the Advanced SIMD and floating-point implementation provides single-precision floating-point instructions. Defined values are:

SIMDSP	Meaning
0000	Not implemented.
0001	Implemented. This value is permitted only if the SIMDInt field is 0001.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

### SIMDInt, bits [15:12]

Advanced SIMD Integer. Indicates whether the Advanced SIMD and floating-point implementation provides integer instructions. Defined values are:

SIMDInt	Meaning
0000	Not implemented.
0001	Implemented.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

**SIMDLS, bits [11:8]**

Advanced SIMD Load/Store. Indicates whether the Advanced SIMD and floating-point implementation provides load/store instructions. Defined values are:

SIMDLS	Meaning
0000	Not implemented.
0001	Implemented.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

**FPDNaN, bits [7:4]**

Default NaN mode. Indicates whether the floating-point implementation provides support only for the Default NaN mode. Defined values are:

FPDNaN	Meaning
0000	Not implemented, or hardware supports only the Default NaN mode.
0001	Hardware supports propagation of NaN values.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

**FPFtZ, bits [3:0]**

Flush to Zero mode. Indicates whether the floating-point implementation provides support only for the Flush-to-Zero mode of operation. Defined values are:

FPFtZ	Meaning
0000	Not implemented, or hardware supports only the Flush-to-Zero mode of operation.
0001	Hardware supports full denormalized number arithmetic.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0001.

**Accessing the MVFR1\_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
MVFR1_EL1	11	000	0000	0011	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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# MVFR2\_EL1, AArch32 Media and VFP Feature Register 2

The MVFR2\_EL1 characteristics are:

## Purpose

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with [MVFR0\\_EL1](#) and [MVFR1\\_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 Floating-point registers functional group.
- The Identification registers functional group.

## Configuration

AArch64 System register MVFR2\_EL1 is architecturally mapped to AArch32 System register [MVFR2](#).

In an implementation where at least one Exception level supports execution in AArch32 state, but there is no support for Advanced SIMD and floating-point operation, this register is RAZ.

In an AArch64-only implementation, this register is UNKNOWN.

## Attributes

MVFR2\_EL1 is a 32-bit register.

## Field descriptions

The MVFR2\_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								
																								FPMisc				SIMDMisc			

### Bits [31:8]

Reserved, RES0.

### FPMisc, bits [7:4]

Indicates whether the floating-point implementation provides support for miscellaneous VFP features.

FPMisc	Meaning
0000	Not implemented, or no support for miscellaneous features.
0001	Support for Floating-point selection.
0010	As 0001, and Floating-point Conversion to Integer with Directed Rounding modes.
0011	As 0010, and Floating-point Round to Integer Floating-point.
0100	As 0011, and Floating-point MaxNum and MinNum.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0100.

**SIMDMisc, bits [3:0]**

Indicates whether the Advanced SIMD implementation provides support for miscellaneous Advanced SIMD features.

SIMDMisc	Meaning
0000	Not implemented, or no support for miscellaneous features.
0001	Floating-point Conversion to Integer with Directed Rounding modes.
0010	As 0001, and Floating-point Round to Integer Floating-point.
0011	As 0010, and Floating-point MaxNum and MinNum.

All other values are reserved.

In ARMv8-A the permitted values are 0000 and 0011.

**Accessing the MVFR2\_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
MVFR2_EL1	11	000	0000	0011	010

**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.

# NZCV, Condition Flags

The NZCV characteristics are:

## Purpose

Allows access to the condition flags.

This register is part of the Process state registers functional group.

## Configuration

There are no configuration notes.

## Attributes

NZCV is a 32-bit register.

## Field descriptions

The NZCV 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
N	Z	C	V	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

### N, bit [31]

Negative condition flag. Set to 1 if the result of the last flag-setting instruction was negative.

### Z, bit [30]

Zero condition flag. Set to 1 if the result of the last flag-setting instruction was zero, and to 0 otherwise. A result of zero often indicates an equal result from a comparison.

### C, bit [29]

Carry condition flag. Set to 1 if the last flag-setting instruction resulted in a carry condition, for example an unsigned overflow on an addition.

### V, bit [28]

Overflow condition flag. Set to 1 if the last flag-setting instruction resulted in an overflow condition, for example a signed overflow on an addition.

### Bits [27:0]

Reserved, RES0.

## Accessing the NZCV

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 &lt;systemreg&gt;, &lt;Xt&gt;

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
NZCV	11	011	0100	0010	000

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# OSDLR\_EL1, OS Double Lock Register

The OSDLR\_EL1 characteristics are:

## Purpose

Used to control the OS Double Lock.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register OSDLR\_EL1 is architecturally mapped to AArch32 System register [DBGOSDLR](#).

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 AArch64. Otherwise, on a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

OSDLR\_EL1 is a 32-bit register.

## Field descriptions

The OSDLR\_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	0	0	DLK

### Bits [31:1]

Reserved, RES0.

### DLK, bit [0]

OS Double Lock control bit. Possible values are:

DLK	Meaning
0	OS Double Lock unlocked.
1	OS Double Lock locked, if <a href="#">DBGPRCR_EL1</a> .CORENPDRQ (Core no powerdown request) bit is set to 0 and the PE is in Non-debug state.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the OSDLR\_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
-------------	-----	-----	-----	-----	-----

OSDLR_EL1	10	000	0001	0011	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.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 :

- If [MDCR\\_EL2.TDOSA](#)==1, Non-secure accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3.TDOSA](#)==1, accesses to this register from EL1 and EL2 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# OSDTRRX\_EL1, OS Lock Data Transfer Register, Receive

The OSDTRRX\_EL1 characteristics are:

## Purpose

Used for save/restore of [DBGDTRRX\\_EL0](#). It is a component of the Debug Communications Channel.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register OSDTRRX\_EL1 is architecturally mapped to AArch32 System register [DBGDTRRXext](#).

## Attributes

OSDTRRX\_EL1 is a 32-bit register.

## Field descriptions

The OSDTRRX\_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
Update DTRRX without side-effect																															

### Bits [31:0]

Update DTRRX without side-effect.

Writes to this register update the value in DTRRX and do not change RXfull.

Reads of this register return the last value written to DTRRX and do not change RXfull.

For the full behavior of the Debug Communications Channel, see 'The Debug Communication Channel and Instruction Transfer Register' in the ARM ARM, chapter H4.

## Accessing the OSDTRRX\_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
OSDTRRX_EL1	10	000	0000	0000	010

## 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.

ARM deprecates reads and writes of OSDTRRX\_EL1 when the OS lock is unlocked.

## 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 :

- 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# OSDTRTX\_EL1, OS Lock Data Transfer Register, Transmit

The OSDTRTX\_EL1 characteristics are:

## Purpose

Used for save/restore of [DBGDTRTX\\_EL0](#). It is a component of the Debug Communications Channel.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register OSDTRTX\_EL1 is architecturally mapped to AArch32 System register [DBGDTRTXext](#).

## Attributes

OSDTRTX\_EL1 is a 32-bit register.

## Field descriptions

The OSDTRTX\_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
Return DTRTX without side-effect																															

### Bits [31:0]

Return DTRTX without side-effect.

Reads of this register return the value in DTRTX and do not change TXfull.

Writes of this register update the value in DTRTX and do not change TXfull.

For the full behavior of the Debug Communications Channel, see 'The Debug Communication Channel and Instruction Transfer Register' in the ARM ARM, chapter H4.

## Accessing the OSDTRTX\_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
OSDTRTX_EL1	10	000	0000	0011	010

## 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.

ARM deprecates reads and writes of OSDTRTX\_EL1 when the OS lock is unlocked.

## 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 :

- 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# OSECCR\_EL1, OS Lock Exception Catch Control Register

The OSECCR\_EL1 characteristics are:

## Purpose

Provides a mechanism for an operating system to access the contents of [EDECCR](#) that are otherwise invisible to software, so it can save/restore the contents of [EDECCR](#) over powerdown on behalf of the external debugger.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register OSECCR\_EL1 is architecturally mapped to AArch32 System register [DBGOSECCR](#).

AArch64 System register OSECCR\_EL1 is architecturally mapped to External register [EDECCR](#).

If [OSLSR\\_EL1](#).OSLK == 0 then OSECCR\_EL1 returns an UNKNOWN value on reads and ignores writes.

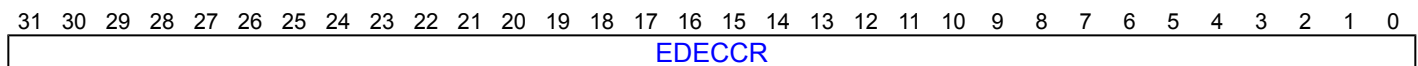
## Attributes

OSECCR\_EL1 is a 32-bit register.

## Field descriptions

The OSECCR\_EL1 bit assignments are:

### When OSLSR.OSLK==1:



### EDECCR, bits [31:0]

Used for save/restore to [EDECCR](#) over powerdown.

Reads or writes to this field are indirect accesses to [EDECCR](#).

## Accessing the OSECCR\_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
OSECCR_EL1	10	000	0000	0110	010

## 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 :

- 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# OSLAR\_EL1, OS Lock Access Register

The OSLAR\_EL1 characteristics are:

## Purpose

Used to lock or unlock the OS lock.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register OSLAR\_EL1 is architecturally mapped to AArch32 System register [DBGOSLAR](#).

AArch64 System register OSLAR\_EL1 is architecturally mapped to External register [OSLAR\\_EL1](#).

## Attributes

OSLAR\_EL1 is a 32-bit register.

## Field descriptions

The OSLAR\_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	0	0	<a href="#">OSLK</a>

### Bits [31:1]

Reserved, RES0.

### OSLK, bit [0]

On writes to OSLAR\_EL1, bit[0] is copied to the OS lock.

Use [OSLSR\\_EL1](#).OSLK to check the current status of the lock.

## Accessing the OSLAR\_EL1

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
OSLAR_EL1	10	000	0001	0000	100

## Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO

x	0	1	-	WO	WO	WO
x	1	1	-	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 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 :

- If [MDCR\\_EL2](#).TDOSA==1, Non-secure write accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3](#).TDOSA==1, write accesses to this register from EL1 and EL2 are trapped to EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# OSLSR\_EL1, OS Lock Status Register

The OSLSR\_EL1 characteristics are:

## Purpose

Provides the status of the OS lock.

This register is part of the Debug registers functional group.

## Configuration

AArch64 System register OSLSR\_EL1 is architecturally mapped to AArch32 System register [DBGOSLSR](#).

This register is in the Cold reset domain. Some or all RW fields of this register have defined reset values. On a Cold reset these apply only if the PE resets into an Exception level that is using AArch64. Otherwise, on a Cold reset RW fields in this register reset to architecturally UNKNOWN values. The register is not affected by a Warm reset.

## Attributes

OSLSR\_EL1 is a 32-bit register.

## Field descriptions

The OSLSR\_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	OSLM[1]	nTT	OSLK	OSLM[0]

### Bits [31:4]

Reserved, RES0.

### OSLM[1], bit [3]

See below for description of the OSLM field.

### nTT, bit [2]

Not 32-bit access. This bit is always RAZ. It indicates that a 32-bit access is needed to write the key to the OS Lock Access Register.

### OSLK, bit [1]

OS Lock Status. The possible values are:

OSLK	Meaning
0	OS lock unlocked.
1	OS lock locked.

The OS lock is locked and unlocked by writing to the OS Lock Access Register.

When this register has an architecturally-defined reset value, this field resets to 1.

### OSLM[0], bit [0]

OS lock model implemented. Identifies the form of OS save and restore mechanism implemented. In ARMv8 these bits are as follows:

OSLM	Meaning
1 0	OS lock implemented. DBGOSSRR not implemented.

All other values are reserved.

## Accessing the OSLSR\_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
OSLSR_EL1	10	000	0001	0001	100

## 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 :

- If [MDCR\\_EL2.TDOSA](#)==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

When EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3.TDOSA](#)==1, read accesses to this register from EL1 and EL2 are trapped to EL3.

# PAN, Privileged Access Never

The PAN characteristics are:

## Purpose

When ARMv8.1-PAN is implemented, allows access to the Privileged Access Never bit.

When ARMv8.1-PAN is not implemented, this register is not implemented.

This register is part of the Process state registers functional group.

## Configuration

This register is introduced in ARMv8.1.

## Attributes

PAN is a 32-bit register.

## Field descriptions

The PAN 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	PAN	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### Bits [31:23]

Reserved, RES0.

### PAN, bit [22]

Privileged Access Never. Defined values are:

PAN	Meaning
0	The translation system is the same as ARMv8.0.
1	Disables privileged read and write accesses to addresses accessible at EL0.

The value of this bit is usually preserved on taking an exception, except in the following situations:

- When the target of the exception is EL1, and the value of the [SCTLR\\_EL1.SPAN](#) bit is 0, this bit is set to 1.
- When the target of the exception is EL2, [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, and the value of the [SCTLR\\_EL2.SPAN](#) bit is 0, this bit is set to 1.

### Bits [21:0]

Reserved, RES0.

## Accessing the PAN

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
PAN	11	000	0100	0010	011

This register can be modified using MSR (immediate) with the following syntax:

MSR <pstatefield>, <imm>

This syntax uses the following encoding in the System instruction encoding space:

<pstatefield>	op0	op1	CRn	op2
PAN	00	000	0100	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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PAR\_EL1, Physical Address Register

The PAR\_EL1 characteristics are:

## Purpose

Returns the output address (OA) from an Address translation instruction that executed successfully, or fault information if the instruction did not execute successfully.

This register is part of the Address translation instructions functional group.

## Configuration

AArch64 System register PAR\_EL1 is architecturally mapped to AArch32 System register [PAR](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

PAR\_EL1 is a 64-bit register.

## Field descriptions

The PAR\_EL1 bit assignments are:

### For all register layouts:

#### F, bit [0]

Indicates whether the instruction performed a successful address translation.

F	Meaning
0	Address translation completed successfully.
1	Address translation aborted.

### When PAR\_EL1.F==0:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32	
ATTR								0	0	0	0	PA[51:48]				PA[47:12]																
PA[47:12]											1	IMP DEF	NS	SH	0	0	0	0	0	0	0	F										
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

This section describes the register value returned by the successful execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

On a successful conversion, the PAR\_EL1 can return a value that indicates the resulting attributes, rather than the values that appear in the translation table descriptors. More precisely:

- The ATTR and SH fields are permitted to report the resulting attributes, as determined by any permitted implementation choices and any applicable configuration bits, instead of reporting the values that appear in the translation table descriptors.
- See the NS bit description for constraints on the value it returns.

#### ATTR, bits [63:56]

Memory attributes for the returned output address. This field uses the same encoding as the Attr<n> fields in [MAIR\\_EL1](#), [MAIR\\_EL2](#), and [MAIR\\_EL3](#).

The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.

### Bits [55:52]

Reserved, RES0.

### PA[51:48], bits [51:48]

In ARMv8.2:

Extension to PA[47:12]. See PA[47:12] for more details.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

### PA[47:12], bits [47:12]

Output address. The output address (OA) corresponding to the supplied input address. This field returns address bits[47:12].

When ARMv8.2-LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, PA[51:48] form the upper part of the address value. Otherwise, for implementations with fewer than 52 physical address bits, the upper bits of this field, corresponding to address bits that are not implemented, are RES0.

### Bit [11]

Reserved, RES1.

### IMP DEF, bit [10]

IMPLEMENTATION DEFINED.

### NS, bit [9]

Non-secure. The NS attribute for a translation table entry from a Secure translation regime.

For a result from a Secure translation regime, this bit reflects the Security state of the physical address space of the translation. This means it reflects the effect of the NSTable bits of earlier levels of the translation table walk if those NSTable bits have an effect on the translation.

For a result from a Non-secure translation regime, this bit is UNKNOWN.

### SH, bits [8:7]

Shareability attribute, for the returned output address. Permitted values are:

SH	Meaning
00	Non-shareable.
10	Outer Shareable.
11	Inner Shareable.

The value 01 is reserved.

---

### Note

This field returns the value 10 for:

- Any type of Device memory.
  - Normal memory with both Inner Non-cacheable and Outer Non-cacheable attributes.
- 

The value returned in this field can be the resulting attribute, as determined by any permitted implementation choices and any applicable configuration bits, instead of the value that appears in the translation table descriptor.



**Bits [6:1]**

Reserved, RES0.

**F, bit [0]**

Indicates whether the instruction performed a successful address translation.

F	Meaning
0	Address translation completed successfully.

**When PAR\_EL1.F==1:**

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
IMP DEF								IMP DEF				IMP DEF				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	1	0	S	PTW	0	FST				F	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

This section describes the register value returned by a fault on the execution of an Address translation instruction. Software might subsequently write a different value to the register, and that write does not affect the operation of the PE.

**IMP DEF, bits [63:56]**

IMPLEMENTATION DEFINED.

**IMP DEF, bits [55:52]**

IMPLEMENTATION DEFINED.

**IMP DEF, bits [51:48]**

IMPLEMENTATION DEFINED.

**Bits [47:12]**

Reserved, RES0.

**Bit [11]**

Reserved, RES1.

**Bit [10]**

Reserved, RES0.

**S, bit [9]**

Indicates the translation stage at which the translation aborted:

S	Meaning
0	Translation aborted because of a fault in the stage 1 translation.
1	Translation aborted because of a fault in the stage 2 translation.

**PTW, bit [8]**

If this bit is set to 1, it indicates the translation aborted because of a stage 2 fault during a stage 1 translation table walk.

**Bit [7]**

Reserved, RES0.

**FST, bits [6:1]**

Fault status code, as shown in the Data Abort ESR encoding.

**F, bit [0]**

Indicates whether the instruction performed a successful address translation.

F	Meaning
1	Address translation aborted.

**Accessing the PAR\_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
PAR_EL1	11	000	0111	0100	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.

# PMCCFILTR\_EL0, Performance Monitors Cycle Count Filter Register

The PMCCFILTR\_EL0 characteristics are:

## Purpose

Determines the modes in which the Cycle Counter, [PMCCNTR\\_EL0](#), increments.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMCCFILTR\_EL0 is architecturally mapped to AArch32 System register [PMCCFILTR](#).

AArch64 System register PMCCFILTR\_EL0 is architecturally mapped to External register [PMCCFILTR\\_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

PMCCFILTR\_EL0 is a 32-bit register.

## Field descriptions

The PMCCFILTR\_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
P	U	NSK	NSU	NSH	M	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

### P, bit [31]

Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are:

P	Meaning
0	Count cycles in EL1.
1	Do not count cycles in EL1.

### U, bit [30]

User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are:

U	Meaning
0	Count cycles in EL0.
1	Do not count cycles in EL0.

### NSK, bit [29]

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of P, cycles in Non-secure EL1 are counted.

Otherwise, cycles in Non-secure EL1 are not counted.

**NSU, bit [28]**

Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of U, cycles in Non-secure EL0 are counted.

Otherwise, cycles in Non-secure EL0 are not counted.

**NSH, bit [27]**

Non-secure EL2 (Hypervisor) filtering bit. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0.

NSH	Meaning
0	Do not count cycles in EL2.
1	Count cycles in EL2.

**M, bit [26]**

Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted.

Otherwise, cycles in Secure EL3 are not counted.

Most applications can ignore this field and set its value to 0.

**Note**

This field is not visible in the AArch32 PMCCFILTR System register.

**Bits [25:0]**

Reserved, RES0.

**Accessing the PMCCFILTR\_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
PMCCFILTR_EL0	11	011	1110	1111	111

PMCCFILTR\_EL0 can also be accessed by using [PMXEVTYPER\\_EL0](#) with [PMSELR\\_EL0](#).SEL set to 0b11111.

**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, 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMCCNTR\_EL0, Performance Monitors Cycle Count Register

The PMCCNTR\_EL0 characteristics are:

## Purpose

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. See 'Time as measured by the Performance Monitors cycle counter' in the ARMv8 ARM, section D5 for more information.

[PMCCFILTR\\_EL0](#) determines the modes and states in which the PMCCNTR\_EL0 can increment.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMCCNTR\_EL0 is architecturally mapped to AArch32 System register [PMCCNTR](#) when accessing as a 64-bit register.

AArch64 System register PMCCNTR\_EL0 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\_EL0 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\_EL0 is a 64-bit register.

## Field descriptions

The PMCCNTR\_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
																CCNT															
																CCNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### CCNT, bits [63:0]

Cycle count. Depending on the values of [PMCR\\_EL0](#).{LC,D}, this field increments in one of the following ways:

- Every processor clock cycle.
- Every 64th processor clock cycle.

Writing 1 to [PMCR\\_EL0](#).C sets this field to 0.

## Accessing the PMCCNTR\_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
PMCCNTR_EL0	11	011	1001	1101	000

## 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.CR](#)==0, and [PMUSERENR\\_EL0.EN](#)==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 :

- 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.

# PMCEID0\_EL0, Performance Monitors Common Event Identification register 0

The PMCEID0\_EL0 characteristics are:

## Purpose

Defines which common architectural and common microarchitectural feature events in the ranges 0x0000 to 0x001F and 0x4000 to 0x401F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

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
ID[16415:16384]																															
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

### ID[16415:16384], bits [63:32]

In ARMv8.2 and ARMv8.1:

PMCEID0\_EL0[63:32] maps to common events 0x4000 to 0x401F. For a list of event numbers and descriptions, see 'Event numbers and mnemonics' in the ARM ARM, section D5.10.

For each bit:

ID[16415:16384]	Meaning
0	The common event is not implemented.
1	The common event is implemented.

Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture.

Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.

In ARMv8.0:

Reserved, RES0.



**ID[31:0], bits [31:0]**

PMCEID0\_EL0[31:0] maps to common events 0x0000 to 0x001F. For a list of event numbers and descriptions, see 'Event numbers and mnemonics' in the ARM ARM, section D5.10.

For each bit:

ID[31:0]	Meaning
0	The common event is not implemented.
1	The common event is implemented.

Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture.

Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.

**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.

# PMCEID1\_EL0, Performance Monitors Common Event Identification register 1

The PMCEID1\_EL0 characteristics are:

## Purpose

Defines which common architectural and common microarchitectural feature events in the ranges 0x0020 to 0x003F and 0x4020 to 0x403F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

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
ID[16447:16416]																															
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

### ID[16447:16416], bits [63:32]

In ARMv8.2 and ARMv8.1:

PMCEID1\_EL0[63:32] maps to common events 0x4020 to 0x403F. For a list of event numbers and descriptions, see 'Event numbers and mnemonics' in the ARM ARM, section D5.10.

For each bit:

ID[16447:16416]	Meaning
0	The common event is not implemented.
1	The common event is implemented.

Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture.

Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.

In ARMv8.0:

Reserved, RES0.

**ID[63:32], bits [31:0]**

PMCEID1\_EL0[31:0] maps to common events 0x0020 to 0x003F. For a list of event numbers and descriptions, see 'Event numbers and mnemonics' in the ARM ARM, section D5.10.

For each bit:

ID[63:32]	Meaning
0	The common event is not implemented.
1	The common event is implemented.

Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture.

Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.

**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.

# PMCNTENCLR\_EL0, Performance Monitors Count Enable Clear register

The PMCNTENCLR\_EL0 characteristics are:

## Purpose

Disables the Cycle Count Register, [PMCCNTR\\_EL0](#), and any implemented event counters [PMEVCNTR<n>](#). Reading this register shows which counters are enabled.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMCNTENCLR\_EL0 is architecturally mapped to AArch32 System register [PMCNTENCLR](#).

AArch64 System register PMCNTENCLR\_EL0 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\_EL0 is a 32-bit register.

## Field descriptions

The PMCNTENCLR\_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
C	P<n>, bit [n]																														

### C, bit [31]

[PMCCNTR\\_EL0](#) 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>\\_EL0](#).

Bits [30:N] are RAZ/WI. When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [MDCR\\_EL2](#).HPMN. Otherwise, N is the value in [PMCR\\_EL0](#).N.

Possible values of each bit are:

P<n>	Meaning
0	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> is disabled. When written, has no effect.
1	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> is enabled. When written, disables <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> .

## Accessing the PMCNTENCLR\_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
PMCNTENCLR_EL0	11	011	1001	1100	010

## 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, 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.

# PMCNTENSET\_EL0, Performance Monitors Count Enable Set register

The PMCNTENSET\_EL0 characteristics are:

## Purpose

Enables the Cycle Count Register, [PMCCNTR\\_EL0](#), and any implemented event counters [PMEVCNTR<n>\\_EL0](#). Reading this register shows which counters are enabled.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMCNTENSET\_EL0 is architecturally mapped to AArch32 System register [PMCNTENSET](#).

AArch64 System register PMCNTENSET\_EL0 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\_EL0 is a 32-bit register.

## Field descriptions

The PMCNTENSET\_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
C	P<n>, bit [n]																														

### C, bit [31]

[PMCCNTR\\_EL0](#) 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>\\_EL0](#).

Bits [30:N] are RAZ/WI. When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [MDCR\\_EL2](#).HPMN. Otherwise, N is the value in [PMCR\\_EL0](#).N.

Possible values of each bit are:

P<n>	Meaning
0	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> is disabled. When written, has no effect.
1	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> event counter is enabled. When written, enables <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> .

## Accessing the PMCNTENSET\_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
PMCNTENSET_EL0	11	011	1001	1100	001

## 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, 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.

# PMCR\_EL0, Performance Monitors Control Register

The PMCR\_EL0 characteristics are:

## Purpose

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMCR\_EL0 is architecturally mapped to AArch32 System register [PMCR](#).

AArch64 System register PMCR\_EL0 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 AArch64. Otherwise, on a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

PMCR\_EL0 is a 32-bit register.

## Field descriptions

The PMCR\_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
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\\_EL0](#) 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\\_EL0](#) is implemented, to 0b11111, which indicates that the [PMCCNTR\\_EL0](#) 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 [MDCR\\_EL2](#).HPMN.

### Bits [10:7]

Reserved, RES0.



**LC, bit [6]**

Long cycle counter enable. Determines which [PMCCNTR\\_EL0](#) bit generates an overflow recorded by [PMOVSr](#)[31].

LC	Meaning
0	Cycle counter overflow on increment that changes <a href="#">PMCCNTR_EL0</a> [31] from 1 to 0.
1	Cycle counter overflow on increment that changes <a href="#">PMCCNTR_EL0</a> [63] from 1 to 0.

ARM deprecates use of  $\text{PMCR\_EL0.LC} = 0$ .

In an AArch64-only implementation, this field 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.

**DP, bit [5]**

Disable cycle counter when event counting is prohibited. The possible values of this bit are:

DP	Meaning
0	<a href="#">PMCCNTR_EL0</a> , if enabled, counts when event counting is prohibited.
1	<a href="#">PMCCNTR_EL0</a> 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 a value that is architecturally UNKNOWN.

**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 a value that is architecturally UNKNOWN.

**D, bit [3]**

Clock divider. The possible values of this bit are:

D	Meaning
0	When enabled, <a href="#">PMCCNTR_EL0</a> counts every clock cycle.
1	When enabled, <a href="#">PMCCNTR_EL0</a> counts once every 64 clock cycles.

In an AArch64-only implementation this field is RES0, otherwise it is an RW field. If  $\text{PMCR\_EL0.LC} = 1$ , this bit is ignored and the cycle counter counts every clock cycle.

ARM deprecates use of  $\text{PMCR\_EL0.D} = 1$ .

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to a value that is architecturally UNKNOWN.

### 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 <a href="#">PMCCNTR_EL0</a> to zero.

This bit is always RAZ.

Resetting [PMCCNTR\\_EL0](#) does not clear the [PMCCNTR\\_EL0](#) 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 <a href="#">PMCCNTR_EL0</a> , 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 [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 <a href="#">PMCCNTR_EL0</a> , are disabled.
1	All counters that are accessible at Non-secure EL1 are enabled by <a href="#">PMCNTESET_EL0</a> .

This bit is RW.

If EL2 is implemented, this bit does not affect the operation of event counters that [MDCR\\_EL2](#).HPMN reserves for EL2 use.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the PMCR\_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
PMCR_EL0	11	011	1001	1100	000

## 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, 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.
- If [MDCR\\_EL2.TPMCR](#)==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.

# PMEVCNTR<n>\_EL0, Performance Monitors Event Count Registers, n = 0 - 30

The PMEVCNTR<n>\_EL0 characteristics are:

## Purpose

Holds event counter n, which counts events, where n is 0 to 30.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMEVCNTR<n>\_EL0 is architecturally mapped to AArch32 System register [PMEVCNTR<n>](#).

AArch64 System register PMEVCNTR<n>\_EL0 is architecturally mapped to External register [PMEVCNTR<n>\\_EL0](#).

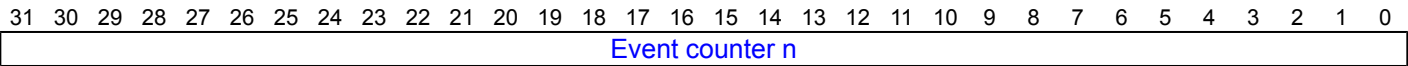
This register is in the Warm reset domain. On a Warm or Cold reset RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

PMEVCNTR<n>\_EL0 is a 32-bit register.

## Field descriptions

The PMEVCNTR<n>\_EL0 bit assignments are:



### Bits [31:0]

Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

## Accessing the PMEVCNTR<n>\_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
PMEVCNTR<n>_EL0	11	011	1110	10:n<4:3>	n<2:0>

PMEVCNTR<n>\_EL0 can also be accessed by using [PMXEVCNTR\\_EL0](#) with [PMSELR\\_EL0](#).SEL set to the value of <n>.

## 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 <n> is greater than or equal to the number of accessible counters, reads and writes of PMEVCNTR<n>\_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- For an access from Non-secure EL1, or an access from Non-secure EL0 when the value of [PMUSERENR\\_EL0.EN](#) is 1, if [PMSELR\\_EL0.SEL](#) 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.

---

#### Note

In an implementation that includes EL2, in Non-secure state at EL0 and EL1, [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 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, 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 :

- 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.

# PMEVTYPER<n>\_EL0, Performance Monitors Event Type Registers, n = 0 - 30

The PMEVTYPER<n>\_EL0 characteristics are:

## Purpose

Configures event counter n, where n is 0 to 30.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMEVTYPER<n>\_EL0 is architecturally mapped to AArch32 System register [PMEVTYPER<n>](#).

AArch64 System register PMEVTYPER<n>\_EL0 is architecturally mapped to External register [PMEVTYPER<n>\\_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

PMEVTYPER<n>\_EL0 is a 32-bit register.

## Field descriptions

The PMEVTYPER<n>\_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
P	U	NSK	NSU	NSH	M	MT	0	0	0	0	0	0	0	0	0	evtCount[15:10]						evtCount[9:0]									

### P, bit [31]

Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are:

P	Meaning
0	Count events in EL1.
1	Do not count events in EL1.

### U, bit [30]

User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are:

U	Meaning
0	Count events in EL0.
1	Do not count events in EL0.

### NSK, bit [29]

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted.

Otherwise, events in Non-secure EL1 are not counted.

**NSU, bit [28]**

Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted.

Otherwise, events in Non-secure EL0 are not counted.

**NSH, bit [27]**

Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0.

NSH	Meaning
0	Do not count events in EL2.
1	Count events in EL2.

**M, bit [26]**

Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted.

Otherwise, cycles in Secure EL3 are not counted.

Most applications can ignore this field and set its value to 0.

**Note**

This field is not visible in the AArch32 PMEVTYPER System register.

**MT, bit [25]**

Multithreading. When the implementation is multi-threaded, the valid values for this bit are:

MT	Meaning
0	Count events only on controlling PE.
1	Count events from any PE with the same affinity at level 1 and above as this PE.

When the implementation is not multi-threaded, this bit is RES0.

**Note**

- An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR\_EL1.MT, when read at the highest implemented Exception level, is 1.
- Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.

**Bits [24:16]**

Reserved, RES0.

**evtCount[15:10], bits [15:10]**

In ARMv8.2 and ARMv8.1:

Extension to evtCount[9:0]. See evtCount[9:0] for more details.

In ARMv8.0:

Reserved, RES0.

**evtCount[9:0], bits [9:0]**

Event to count. The event number of the event that is counted by event counter [PMEVCNTR<n>\\_EL0](#).

Software must program this field with an event that is supported by the PE being programmed.

There are three ranges of event numbers:

- Event numbers in the range 0x000 to 0x03F are common architectural and microarchitectural events.
- Event numbers in the range 0x040 to 0x0BF are ARM recommended common architectural and microarchitectural events.
- Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events.

If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type:

- For the range 0x000 to 0x03F, no events are counted, and the value returned by a direct or external read of the evtCount field is the value written to the field.
- For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the evtCount field is UNKNOWN.

**Note**

UNPREDICTABLE means the event must not expose privileged information.

ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

**Accessing the PMEVTYPER<n>\_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
PMEVTYPER<n>_EL0	11	011	1110	11:n<4:3>	n<2:0>

**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.

PMEVTYPER<n>\_EL0 can also be accessed by using [PMXEVTYPER\\_EL0](#) with [PMSELR\\_EL0](#).SEL set to n.

If <n> is greater than or equal to the number of accessible counters, reads and writes of PMEVTYPER<n>\_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- For an access from Non-secure EL1, or an access from Non-secure EL0 when the value of [PMUSERENR\\_EL0](#).EN is 1, if [PMSELR\\_EL0](#).SEL 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.



---

**Note**

In an implementation that includes EL2, in Non-secure state at EL0 and EL1, [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 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, 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMINTENCLR\_EL1, Performance Monitors Interrupt Enable Clear register

The PMINTENCLR\_EL1 characteristics are:

## Purpose

Disables the generation of interrupt requests on overflows from the Cycle Count Register, [PMCCNTR\\_EL0](#), and the event counters [PMEVCNTR<n>\\_EL0](#). Reading the register shows which overflow interrupt requests are enabled.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMINTENCLR\_EL1 is architecturally mapped to AArch32 System register [PMINTENCLR](#).

AArch64 System register PMINTENCLR\_EL1 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\_EL1 is a 32-bit register.

## Field descriptions

The PMINTENCLR\_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
C	P<n>, bit [n]																														

### C, bit [31]

[PMCCNTR\\_EL0](#) 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>\\_EL0](#).

When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [MDCR\\_EL2](#).HPMN. Otherwise, N is the value in [PMCR\\_EL0](#).N.

Bits [30:N] are RAZ/WI.

Possible values are:

P<n>	Meaning
0	When read, means that the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> event counter interrupt request is disabled. When written, has no effect.
1	When read, means that the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> event counter interrupt request is enabled. When written, disables the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> interrupt request.

## Accessing the PMINTENCLR\_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
PMINTENCLR_EL1	11	000	1001	1110	010

## 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.

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 EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3.TPM](#)=1, accesses to this register from EL1 and EL2 are trapped to EL3.

# PMINTENSET\_EL1, Performance Monitors Interrupt Enable Set register

The PMINTENSET\_EL1 characteristics are:

## Purpose

Enables the generation of interrupt requests on overflows from the Cycle Count Register, [PMCCNTR\\_EL0](#), and the event counters [PMEVCNTR<n>\\_EL0](#). Reading the register shows which overflow interrupt requests are enabled.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMINTENSET\_EL1 is architecturally mapped to AArch32 System register [PMINTENSET](#).

AArch64 System register PMINTENSET\_EL1 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\_EL1 is a 32-bit register.

## Field descriptions

The PMINTENSET\_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
C	P<n>, bit [n]																														

### C, bit [31]

[PMCCNTR\\_EL0](#) 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>\\_EL0](#).

When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [MDCR\\_EL2](#).HPMN. Otherwise, N is the value in [PMCR\\_EL0](#).N.

Bits [30:N] are RAZ/WI.

Possible values are:

P<n>	Meaning
0	When read, means that the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> event counter interrupt request is disabled. When written, has no effect.
1	When read, means that the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> event counter interrupt request is enabled. When written, enables the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> interrupt request.

## Accessing the PMINTENSET\_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
PMINTENSET_EL1	11	000	1001	1110	001

## 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.

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 EL3 is implemented and is using AArch64 :

- If [MDCR\\_EL3.TPM](#)==1, accesses to this register from EL1 and EL2 are trapped to EL3.

# PMOVSLR\_EL0, Performance Monitors Overflow Flag Status Clear Register

The PMOVSLR\_EL0 characteristics are:

## Purpose

Contains the state of the overflow bit for the Cycle Count Register, [PMCCNTR\\_EL0](#), and each of the implemented event counters [PMEVCNTR<n>](#). Writing to this register clears these bits.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMOVSLR\_EL0 is architecturally mapped to AArch32 System register [PMOVSr](#).

AArch64 System register PMOVSLR\_EL0 is architecturally mapped to External register [PMOVSLR\\_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

PMOVSLR\_EL0 is a 32-bit register.

## Field descriptions

The PMOVSLR\_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
C	P<n>, bit [n]																														

### C, bit [31]

[PMCCNTR\\_EL0](#) 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\\_EL0](#).LC controls whether an overflow is detected from [PMCCNTR\\_EL0](#)[31] or from [PMCCNTR\\_EL0](#).

### P<n>, bit [n], for n = 0 to 30

Event counter overflow clear bit for [PMEVCNTR<n>\\_EL0](#).

Bits [30:N] are RAZ/WI.

When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [MDCR\\_EL2](#).HPMN. Otherwise, N is the value in [PMCR\\_EL0](#).N.

Possible values of each bit are:

P<n>	Meaning
0	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> has not overflowed. When written, has no effect.
1	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> has overflowed. When written, clears the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> overflow bit to 0.

## Accessing the PMOVSCLR\_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
PMOVSCLR_EL0	11	011	1001	1100	011

## 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, 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.

# PMOVSSET\_EL0, Performance Monitors Overflow Flag Status Set register

The PMOVSSET\_EL0 characteristics are:

## Purpose

Sets the state of the overflow bit for the Cycle Count Register, [PMCCNTR\\_EL0](#), and each of the implemented event counters [PMEVCNTR<n>](#).

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMOVSSET\_EL0 is architecturally mapped to AArch32 System register [PMOVSSET](#).

AArch64 System register PMOVSSET\_EL0 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\_EL0 is a 32-bit register.

## Field descriptions

The PMOVSSET\_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
C	P<n>, bit [n]																														

### C, bit [31]

[PMCCNTR\\_EL0](#) 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>\\_EL0](#).

Bits [30:N] are RAZ/WI.

When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [MDCR\\_EL2](#).HPMN. Otherwise, N is the value in [PMCR\\_EL0](#).N.

Possible values are:

P<n>	Meaning
0	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> has not overflowed. When written, has no effect.
1	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> has overflowed. When written, sets the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> overflow bit to 1.



## Accessing the PMOVSSET\_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
PMOVSSET_EL0	11	011	1001	1110	011

## 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, 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.

# 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, the results of access to [PMXEVTYPER\\_EL0](#) or [PMXVCNTR\\_EL0](#) are CONSTRAINED UNPREDICTABLE, and can be one of the following:

- Access to [PMXEVTYPER\\_EL0](#) or [PMXVCNTR\\_EL0](#) is UNDEFINED.
- Access to [PMXEVTYPER\\_EL0](#) or [PMXVCNTR\\_EL0](#) behaves as a NOP.

- Access to [PMXEVTYPER\\_EL0](#) or [PMXEVCNTR\\_EL0](#) behaves as if the register is RAZ/WI.
- Access to [PMXEVTYPER\\_EL0](#) or [PMXEVCNTR\\_EL0](#) behaves as if the PMSELR\_EL0.SEL field contains an UNKNOWN value.
- Access to [PMXEVTYPER\\_EL0](#) or [PMXEVCNTR\\_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.

# PMSWINC\_EL0, Performance Monitors Software Increment register

The PMSWINC\_EL0 characteristics are:

## Purpose

Increments a counter that is configured to count the Software increment event, event 0x00. For more information, see 'SW\_INCR' in the ARMv8 ARM, section D5.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMSWINC\_EL0 is architecturally mapped to AArch32 System register [PMSWINC](#).

AArch64 System register PMSWINC\_EL0 is architecturally mapped to External register [PMSWINC\\_EL0](#).

## Attributes

PMSWINC\_EL0 is a 32-bit register.

## Field descriptions

The PMSWINC\_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	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>\\_EL0](#).

Bits [30:N] are WL.

When EL2 is implemented, in Non-secure EL1 and EL0, N is the value in [MDCR\\_EL2](#).HPMN. 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 <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> is enabled and configured to count the software increment event, increments <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> by 1. If <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> is disabled, or not configured to count the software increment event, the write to this bit is ignored.

## Accessing the PMSWINC\_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 &lt;systemreg&gt;, &lt;Xt&gt;

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
PMSWINC_EL0	11	011	1001	1100	100

## 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](#).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 :

- If [MDCR\\_EL2](#).TPM==1, Non-secure write 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, write accesses to this register from EL0, EL1, and EL2 are trapped to EL3.

# PMUSERENR\_EL0, Performance Monitors User Enable Register

The PMUSERENR\_EL0 characteristics are:

## Purpose

Enables or disables EL0 access to the Performance Monitors.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMUSERENR\_EL0 is architecturally mapped to AArch32 System register [PMUSERENR](#).

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

PMUSERENR\_EL0 is a 32-bit register.

## Field descriptions

The PMUSERENR\_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	0	ER	CR	SW	EN

### Bits [31:4]

Reserved, RES0.

### ER, bit [3]

Event counter read trap control:

ER	Meaning
0	EL0 using AArch64: EL0 reads of the <a href="#">PMXEVNTR_EL0</a> and <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> , and EL0 read/write accesses to the <a href="#">PMSELR_EL0</a> , are trapped to EL1 if PMUSERENR_EL0.EN is also 0. EL0 using AArch32: EL0 reads of the <a href="#">PMXEVNTR</a> and <a href="#">PMEVCNTR&lt;n&gt;</a> , and EL0 read/write accesses to the <a href="#">PMSELR</a> , are trapped to EL1 if PMUSERENR_EL0.EN is also 0.
1	This control does not cause any instructions to be trapped.

### CR, bit [2]

Cycle counter read trap control:

CR	Meaning
0	EL0 using AArch64: EL0 read accesses to the <a href="#">PMCCNTR_EL0</a> are trapped to EL1 if PMUSERENR_EL0.EN is also 0. EL0 using AArch32: EL0 read accesses to the <a href="#">PMCCNTR</a> are trapped to EL1 if PMUSERENR_EL0.EN is also 0.
1	This control does not cause any instructions to be trapped.

**SW, bit [1]**

Software Increment write trap control:

SW	Meaning
0	EL0 using AArch64: EL0 writes to the <a href="#">PMSWINC_EL0</a> are trapped to EL1 if PMUSERENR_EL0.EN is also 0. EL0 using AArch32: EL0 writes to the <a href="#">PMSWINC</a> are trapped to EL1 if PMUSERENR_EL0.EN is also 0.
1	This control does not cause any instructions to be trapped.

**EN, bit [0]**

Traps EL0 accesses to the Performance Monitors registers to EL1, from both Execution states:

EN	Meaning
0	EL0 accesses to the Performance Monitors registers are trapped to EL1, unless enabled by one of PMUSERENR_EL0.{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\_EL0 and [PMUSERENR](#) registers are always RO at EL0 and not trapped by this bit.
- EL0 cannot read or write [PMINTENSET\\_EL1](#) and [PMINTENCLR\\_EL1](#), or [PMINTENSET](#) and [PMINTENCLR](#).

**Accessing the PMUSERENR\_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
PMUSERENR_EL0	11	011	1001	1110	000

**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 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 :

- 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# PMXEVCNTR\_EL0, Performance Monitors Selected Event Count Register

The PMXEVCNTR\_EL0 characteristics are:

## Purpose

Reads or writes the value of the selected event counter, [PMEVCNTR<n>\\_EL0](#). [PMSELR\\_EL0](#).SEL determines which event counter is selected.

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMXEVCNTR\_EL0 is architecturally mapped to AArch32 System register [PMXEVCNTR](#).

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\_EL0 is a 32-bit register.

## Field descriptions

The PMXEVCNTR\_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
<a href="#">PMEVCNTR&lt;n&gt;</a>																															

### PMEVCNTR<n>, bits [31:0]

Value of the selected event counter, [PMEVCNTR<n>\\_EL0](#), where n is the value stored in [PMSELR\\_EL0](#).SEL.

## Accessing the PMXEVCNTR\_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
PMXEVCNTR_EL0	11	011	1001	1101	010

## 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\\_EL0.SEL](#) is greater than or equal to the number of accessible counters then reads and writes of PMXEVCNTR\_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if [PMSELR\\_EL0.SEL](#) has an UNKNOWN value less than the number of counters accessible at the current Exception level and Security state.
- Accesses to the register behave as if [PMSELR\\_EL0.SEL](#) is 31.
- For an access from Non-secure EL1, or an access from Non-secure EL0 when the value of [PMUSERENR\\_EL0.EN](#) is 1, if [PMSELR\\_EL0.SEL](#) 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.

#### Note

In an implementation that includes EL2, in Non-secure state at EL0 and EL1, [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 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, 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 :

- 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.

# PMXEVTYPER\_EL0, Performance Monitors Selected Event Type Register

The PMXEVTYPER\_EL0 characteristics are:

## Purpose

When [PMSELR\\_EL0.SEL](#) selects an event counter, this accesses a [PMEVTYPER<n>\\_EL0](#) register. When [PMSELR\\_EL0.SEL](#) selects the cycle counter, this accesses [PMCCFILTR\\_EL0](#).

This register is part of the Performance Monitors registers functional group.

## Configuration

AArch64 System register PMXEVTYPER\_EL0 is architecturally mapped to AArch32 System register [PMXEVTYPER](#).

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

PMXEVTYPER\_EL0 is a 32-bit register.

## Field descriptions

The PMXEVTYPER\_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
Event type register or PMCCFILTR_EL0																															

### Bits [31:0]

Event type register or [PMCCFILTR\\_EL0](#).

When [PMSELR\\_EL0.SEL](#) == 31, this register accesses [PMCCFILTR\\_EL0](#).

Otherwise, this register accesses [PMEVTYPER<n>\\_EL0](#) where n is the value in [PMSELR\\_EL0.SEL](#).

## Accessing the PMXEVTYPER\_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
PMXEVTYPER_EL0	11	011	1001	1101	001

## 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\\_EL0.SEL](#) is greater than or equal to the number of accessible counters then reads and writes of PMXEVTYPER\_EL0 are CONSTRAINED UNPREDICTABLE, and the following behaviors are permitted:

- Accesses to the register are UNDEFINED.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a NOP.
- Accesses to the register behave as if [PMSELR\\_EL0.SEL](#) has an UNKNOWN value less than the number of counters accessible at the current Exception level and Security state.
- Accesses to the register behave as if [PMSELR\\_EL0.SEL](#) is 31.
- For an access from Non-secure EL1, or an access from Non-secure EL0 when the value of [PMUSERENR\\_EL0.EN](#) is 1, if [PMSELR\\_EL0.SEL](#) 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.

---

#### Note

In an implementation that includes EL2, in Non-secure state at EL0 and EL1, [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 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, 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.

# REVIDR\_EL1, Revision ID Register

The REVIDR\_EL1 characteristics are:

## Purpose

Provides implementation-specific minor revision information.

This register is part of the Identification registers functional group.

## Configuration

AArch64 System register REVIDR\_EL1 is architecturally mapped to AArch32 System register [REVIDR](#).

If REVIDR\_EL1 has the same value as [MIDR\\_EL1](#), then its contents have no significance.

## Attributes

REVIDR\_EL1 is a 32-bit register.

## Field descriptions

The REVIDR\_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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

## Accessing the REVIDR\_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
REVIDR_EL1	11	000	0000	0000	110

## 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](#).TID1==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](#).TID1==1, Non-secure read accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# RMR\_EL1, Reset Management Register (EL1)

The RMR\_EL1 characteristics are:

## Purpose

If EL1 is the highest implemented Exception level and this register is implemented:

- A write to the register at EL1 can request a Warm reset.
- If EL1 can use AArch32 and AArch64, this register specifies the Execution state that the PE boots into on a Warm reset.

This register is part of the Reset management registers functional group.

## Configuration

AArch64 System register RMR\_EL1 is architecturally mapped to AArch32 System register [RMR](#) when EL1 is highest implemented Exception level.

Only implemented if EL1 is the highest implemented Exception level. In this case:

- If EL1 can use AArch32 and AArch64 then this register must be implemented.
- If EL1 cannot use AArch32 then it is IMPLEMENTATION DEFINED whether the register is implemented.

See the field descriptions for the reset values. These apply whenever the register is implemented.

## Attributes

RMR\_EL1 is a 32-bit register.

## Field descriptions

The RMR\_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	0	RR	AA64

### Bits [31:2]

Reserved, RES0.

### RR, bit [1]

Reset Request. Setting this bit to 1 requests a Warm reset.

This field resets to 0 on a Warm or Cold reset.

### AA64, bit [0]

When EL1 can use AArch32, determines which Execution state the PE boots into after a Warm reset:

AA64	Meaning
0	AArch32.
1	AArch64.

On coming out of the Warm reset, execution starts at the IMPLEMENTATION DEFINED reset vector address of the specified Execution state.

If EL1 cannot use AArch32 this bit is RAO/WI.

When implemented as a RW field, this field resets to 1 on a Cold reset.

## Accessing the RMR\_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
RMR_EL1	11	000	1100	0000	010

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
EL1 is the highest implemented Exception level	x	x	x	-	RW	n/a	n/a

This table applies to all instructions that can access this register.

When RMR\_EL1 is not implemented, the encoding for this register is UNDEFINED.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# RMR\_EL2, Reset Management Register (EL2)

The RMR\_EL2 characteristics are:

## Purpose

If EL2 is the highest implemented Exception level and this register is implemented:

- A write to the register at EL2 can request a Warm reset.
- If EL2 can use AArch32 and AArch64, this register specifies the Execution state that the PE boots into on a Warm reset.

This register is part of:

- The Virtualization registers functional group.
- The Reset management registers functional group.

## Configuration

AArch64 System register RMR\_EL2 is architecturally mapped to AArch32 System register [HRMR](#).

Only implemented if EL2 is the highest implemented Exception level. In this case:

- If EL2 can use AArch32 and AArch64 then this register must be implemented.
- If EL2 cannot use AArch32 then it is IMPLEMENTATION DEFINED whether the register is implemented.

See the field descriptions for the reset values. These apply whenever the register is implemented.

## Attributes

RMR\_EL2 is a 32-bit register.

## Field descriptions

The RMR\_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	
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	RR	AA64

### Bits [31:2]

Reserved, RES0.

### RR, bit [1]

Reset Request. Setting this bit to 1 requests a Warm reset.

This field resets to 0 on a Warm or Cold reset.

### AA64, bit [0]

When EL2 can use AArch32, determines which Execution state the PE boots into after a Warm reset:

AA64	Meaning
0	AArch32.
1	AArch64.

On coming out of the Warm reset, execution starts at the IMPLEMENTATION DEFINED reset vector address of the specified Execution state.

If EL2 cannot use AArch32 this bit is RAO/WI.

When implemented as a RW field, this field resets to 1 on a Cold reset.

## Accessing the RMR\_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
RMR_EL2	11	100	1100	0000	010

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
EL2 is the highest implemented Exception level	x	0	1	-	-	RW	n/a
EL2 is the highest implemented Exception level	x	1	1	-	n/a	RW	n/a

This table applies to all instructions that can access this register.

When RMR\_EL2 is not implemented, the encoding for this register is UNDEFINED.

# RMR\_EL3, Reset Management Register (EL3)

The RMR\_EL3 characteristics are:

## Purpose

If EL3 is the implemented and this register is implemented:

- A write to the register at EL3 can request a Warm reset.
- If EL3 can use AArch32 and AArch64, this register specifies the Execution state that the PE boots into on a Warm reset.

This register is part of the Reset management registers functional group.

## Configuration

AArch64 System register RMR\_EL3 is architecturally mapped to AArch32 System register [RMR](#) when EL3 is implemented.

When EL3 is implemented:

- If EL3 can use AArch32 and AArch64 then this register must be implemented.
- If EL3 cannot use AArch32 then it is IMPLEMENTATION DEFINED whether the register is implemented.

See the field descriptions for the reset values. These apply whenever the register is implemented.

## Attributes

RMR\_EL3 is a 32-bit register.

## Field descriptions

The RMR\_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
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	RR	AA64

### Bits [31:2]

Reserved, RES0.

### RR, bit [1]

Reset Request. Setting this bit to 1 requests a Warm reset.

This field resets to 0 on a Warm or Cold reset.

### AA64, bit [0]

When EL3 can use AArch32, determines which Execution state the PE boots into after a Warm reset:

AA64	Meaning
0	AArch32.
1	AArch64.

On coming out of the Warm reset, execution starts at the IMPLEMENTATION DEFINED reset vector address of the specified Execution state.

If EL3 cannot use AArch32 this bit is RAO/WI.

When implemented as a RW field, this field resets to 1 on a Cold reset.

## Accessing the RMR\_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
RMR_EL3	11	110	1100	0000	010

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
EL3 is the highest implemented Exception level	x	x	0	-	-	n/a	RW
EL3 is the highest implemented Exception level	x	0	1	-	-	-	RW
EL3 is the highest implemented Exception level	x	1	1	-	n/a	-	RW

This table applies to all instructions that can access this register.

When RMR\_EL3 is not implemented, the encoding for this register is UNDEFINED.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# RVBAR\_EL1, Reset Vector Base Address Register (if EL2 and EL3 not implemented)

The RVBAR\_EL1 characteristics are:

## Purpose

If EL1 is the highest Exception level implemented, contains the IMPLEMENTATION DEFINED address that execution starts from after reset when executing in AArch64 state.

This register is part of the Reset management registers functional group.

## Configuration

Only implemented if the highest Exception level implemented is EL1.

## Attributes

RVBAR\_EL1 is a 64-bit register.

## Field descriptions

The RVBAR\_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
Reset Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset Address																															

### Bits [63:0]

Reset Address. The IMPLEMENTATION DEFINED address that execution starts from after reset when executing in 64-bit state. Bits[1:0] of this register are 00, as this address must be aligned, and the address must be within the physical address size supported by the PE.

## Accessing the RVBAR\_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
RVBAR_EL1	11	000	1100	0000	001

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
EL1 is the highest implemented Exception level	x	x	x	-	RO	n/a	n/a

This table applies to all instructions that can access this register.

When EL1 is not the highest implemented Exception level, the encoding for this register is UNDEFINED.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# RVBAR\_EL2, Reset Vector Base Address Register (if EL3 not implemented)

The RVBAR\_EL2 characteristics are:

## Purpose

If EL2 is the highest Exception level implemented, contains the IMPLEMENTATION DEFINED address that execution starts from after reset when executing in AArch64 state.

This register is part of the Reset management registers functional group.

## Configuration

Only implemented if the highest Exception level implemented is EL2.

## Attributes

RVBAR\_EL2 is a 64-bit register.

## Field descriptions

The RVBAR\_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
Reset Address																															
Reset Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Reset Address. The IMPLEMENTATION DEFINED address that execution starts from after reset when executing in 64-bit state. Bits[1:0] of this register are 00, as this address must be aligned, and the address must be within the physical address size supported by the PE.

## Accessing the RVBAR\_EL2

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
RVBAR_EL2	11	100	1100	0000	001

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
EL2 is the highest implemented Exception level	x	0	1	-	-	RO	n/a
EL2 is the highest implemented Exception level	x	1	1	-	n/a	RO	n/a

This table applies to all instructions that can access this register.

When EL2 is not the highest implemented Exception level, the encoding for this register is UNDEFINED.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# RVBAR\_EL3, Reset Vector Base Address Register (if EL3 implemented)

The RVBAR\_EL3 characteristics are:

## Purpose

If EL3 is the highest Exception level implemented, contains the IMPLEMENTATION DEFINED address that execution starts from after reset when executing in AArch64 state.

This register is part of the Reset management registers functional group.

## Configuration

Only implemented if the highest Exception level implemented is EL3.

## Attributes

RVBAR\_EL3 is a 64-bit register.

## Field descriptions

The RVBAR\_EL3 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
Reset Address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset Address																															

### Bits [63:0]

Reset Address. The IMPLEMENTATION DEFINED address that execution starts from after reset when executing in 64-bit state. Bits[1:0] of this register are 00, as this address must be aligned, and the address must be within the physical address size supported by the PE.

## Accessing the RVBAR\_EL3

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
RVBAR_EL3	11	110	1100	0000	001

## Accessibility

The register is accessible as follows:

Configuration	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
EL3 is the highest implemented Exception level	x	x	0	-	-	n/a	RO
EL3 is the highest implemented Exception level	x	0	1	-	-	-	RO
EL3 is the highest implemented Exception level	x	1	1	-	n/a	-	RO

This table applies to all instructions that can access this register.

When EL3 is not the highest implemented Exception level, the encoding for this register is UNDEFINED.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# S3\_<op1>\_<Cn>\_<Cm>\_<op2>, IMPLEMENTATION DEFINED registers

The S3\_<op1>\_<Cn>\_<Cm>\_<op2> characteristics are:

## Purpose

This area of the instruction set space is reserved for IMPLEMENTATION DEFINED registers.

This register is part of the IMPLEMENTATION DEFINED functional group.

## Configuration

There are no configuration notes.

## Attributes

S3\_<op1>\_<Cn>\_<Cm>\_<op2> is a 64-bit register.

## Field descriptions

The S3\_<op1>\_<Cn>\_<Cm>\_<op2> 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
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

## Accessing the S3\_<op1>\_<Cn>\_<Cm>\_<op2>

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
S3_<op1>_C<Cn>_C<Cm>_<op2>	11	op1<2:0>	Cn<3:0>	Cm<3:0>	op2<2:0>

The value of <Cn> must be either 11 or 15. Other values may refer to architecturally-defined registers.

## Accessibility

The accessibility of registers with these encodings is IMPLEMENTATION DEFINED.

## 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](#).TIDCP==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](#).TIDCP==1, Non-secure accesses to this register from EL1 are trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# SCR\_EL3, Secure Configuration Register

The SCR\_EL3 characteristics are:

## Purpose

Defines the configuration of the current Security state. It specifies:

- The Security state of EL0 and EL1, either Secure or Non-secure.
- The Execution state at lower Exception levels.
- Whether IRQ, FIQ, and External Abort interrupts are taken to EL3.

This register is part of the Security registers functional group.

## Configuration

AArch64 System register SCR\_EL3 can be mapped to AArch32 System register [SCR](#), but this is not architecturally mandated.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

SCR\_EL3 is a 32-bit register.

## Field descriptions

The SCR\_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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	TERR	TLOR	TWET	TWIST	RWSIF	HCE	SMD	0	1	1	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	Does not trap accesses to record registers from EL1 and EL2 to EL3.
1	Accesses to the ER* registers from EL1 and EL2 generate a Trap exception to EL3.

This bit resets to 0 on Warm reset.

When the RAS Extension is not implemented, this field is RES0.

### TLOR, bit [14]

In 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 EL1 and EL2 to EL3, unless the access has been trapped to EL2.

TLOR	Meaning
0	This control does not cause any instructions to be trapped.
1	EL1 and EL2 accesses to the LOR registers that are not UNDEFINED are trapped to EL3, unless it is trapped <a href="#">HCR_EL2.TLOR</a> .

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.

### TWE, bit [13]

Traps EL2, EL1, and EL0 execution of WFE instructions to EL3, from both Security states and 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 any Exception level lower than EL3 is trapped to EL3, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by <a href="#">SCTLR.nTWE</a> , <a href="#">HCR.TWE</a> , <a href="#">SCTLR_EL1.nTWE</a> , <a href="#">SCTLR_EL2.nTWE</a> , or <a href="#">HCR_EL2.TWE</a> .

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.

### TWI, bit [12]

Traps EL2, EL1, and EL0 execution of WFI instructions to EL3, from both Security states and 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 any Exception level lower than EL3 is trapped to EL3, if the instruction would otherwise have caused the PE to enter a low-power state and it is not trapped by <a href="#">SCTLR.nTWI</a> , <a href="#">HCR.TWI</a> , <a href="#">SCTLR_EL1.nTWI</a> , <a href="#">SCTLR_EL2.nTWI</a> , or <a href="#">HCR_EL2.TWI</a> .

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.

### ST, bit [11]

Traps Secure EL1 accesses to the Counter-timer Physical Secure timer registers to EL3, from AArch64 state only.

ST	Meaning
0	Secure EL1 using AArch64 accesses to the <a href="#">CNTPS_TVAL_EL1</a> , <a href="#">CNTPS_CTL_EL1</a> , and <a href="#">CNTPS_CVAL_EL1</a> are trapped to EL3.
1	This control does not cause any instructions to be trapped.

### RW, bit [10]

Execution state control for lower Exception levels.

RW	Meaning
0	Lower levels are all AArch32.
1	The next lower level is AArch64. If EL2 is present: <ul style="list-style-type: none"> <li>EL2 is AArch64.</li> <li>EL2 controls EL1 and EL0 behaviors.</li> </ul> If EL2 is not present: <ul style="list-style-type: none"> <li>EL1 is AArch64.</li> <li>EL0 is determined by the Execution state described in the current process state when executing at EL0.</li> </ul>

If all lower Exception levels cannot use AArch32 then this bit is RAO/WI.

This bit is permitted to be cached in a TLB.

#### 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.

#### HCE, bit [8]

Hypervisor Call instruction enable. Enables HVC instructions at EL3, EL2, and Non-secure EL1, in both Execution states.

HCE	Meaning
0	HVC instructions are UNDEFINED at EL3, EL2, and Non-secure EL1, and any resulting exception is taken from the current Exception level to the current Exception level.
1	HVC instructions are enabled at EL1 and above.

#### Note

HVC instructions are always UNDEFINED at EL0.

If EL2 is not implemented, this bit is RES0.

#### SMD, bit [7]

Secure Monitor Call disable. Disables SMC instructions at EL1 and above, from both Security states and both Execution states.

SMD	Meaning
0	SMC instructions are enabled at EL1 and above.
1	SMC instructions are UNDEFINED at EL1 and above.

#### Note

SMC instructions are always UNDEFINED at EL0.

#### Bit [6]

Reserved, RES0.

#### Bits [5:4]

Reserved, RES1.

**EA, bit [3]**

External Abort and SError Interrupt Routing.

EA	Meaning
0	When executing at Exception levels below EL3, External Aborts and SError Interrupts are not taken to EL3. In addition, when executing at EL3: <ul style="list-style-type: none"> <li>• SError Interrupts are not taken.</li> <li>• External Aborts are taken to EL3.</li> </ul>
1	When executing at any Exception level, External Aborts and SError Interrupts are taken to EL3.

For more information, see 'Asynchronous exception routing' in the ARMv8 ARM, section D1 (The AArch64 System Level Programmers' Model).

**FIQ, bit [2]**

Physical FIQ Routing.

FIQ	Meaning
0	When executing at Exception levels below EL3, physical FIQ interrupts are not taken to EL3. When executing at EL3, physical FIQ interrupts are not taken.
1	When executing at any Exception level, physical FIQ interrupts are taken to EL3.

For more information, see 'Asynchronous exception routing' in the ARMv8 ARM, section D1.

**IRQ, bit [1]**

Physical IRQ Routing.

IRQ	Meaning
0	When executing at Exception levels below EL3, physical IRQ interrupts are not taken to EL3. When executing at EL3, physical IRQ interrupts are not taken.
1	When executing at any Exception level, physical IRQ interrupts are taken to EL3.

For more information, see 'Asynchronous exception routing' in the ARMv8 ARM, section D1.

**NS, bit [0]**

Non-secure bit.

NS	Meaning
0	Indicates that EL0 and EL1 are in Secure state, and so memory accesses from those Exception levels can access Secure memory. When executing at EL3: <ul style="list-style-type: none"> <li>• The <a href="#">AT S1E2R</a>, <a href="#">AT S1E2W</a>, <a href="#">TLBI VAE2</a>, <a href="#">TLBI VALE2</a>, <a href="#">TLBI VAE2IS</a>, <a href="#">TLBI VALE2IS</a>, <a href="#">TLBI ALLE2</a>, and <a href="#">TLBI ALLE2IS</a> System instructions are UNDEFINED.</li> <li>• Each AT S1E** System instruction executes as the corresponding AT S1E** instruction. For example, <a href="#">AT S1E0R</a> executes as <a href="#">AT S1E0R</a>.</li> <li>• Each of the <a href="#">TLBI IPAS2E1</a>, <a href="#">TLBI IPAS2E1IS</a>, <a href="#">TLBI IPAS2LE1</a>, and <a href="#">TLBI IPAS2LE1IS</a> System instructions executes as a NOP.</li> <li>• A <a href="#">TLBI VMALLS12E1</a> System instruction executes as <a href="#">TLBI VMALLE1</a>, and a <a href="#">TLBI VMALLS12E1IS</a> System instruction executes as <a href="#">TLBI VMALLE1IS</a>.</li> </ul>
1	Indicates that EL0 and EL1 are in Non-secure state, and so memory accesses from those Exception levels cannot access Secure memory.

**Note**



EL2 is not supported in the Secure state. When SCR\_EL3.NS==0, it is not possible to enter EL2, and the EL2 state has no effect on execution. See 'Virtualization' in the ARMv8 ARM, section D1.5.

## Accessing the SCR\_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
SCR_EL3	11	110	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
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.

# 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:

3130	29	28	27	26	25	24	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	LSMAOE	n	TL	SMD	0	UCI	EE	E0E	SPAN	1	IESB	1	WXN	n	TWE	0	n	TWI	UCT	DZE	0	1	1	0	UMASED	ITD	0	CP15	BEN	SA0	SACAM

### Bits [31:30]

Reserved, RES0.

### LSMAOE, bit [29]

In 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.

In ARMv8.1 and ARMv8.0:

Reserved, RES1.

**nTLSMD, bit [28]**

In 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.

In ARMv8.1 and ARMv8.0:

Reserved, RES1.

**Bit [27]**

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 <a href="#">DC CVAU</a> , <a href="#">DC CIVAC</a> , <a href="#">DC CVAC</a> , <a href="#">DC CVAP</a> , or <a href="#">IC IVAU</a> 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.

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:

<b>E0E</b>	<b>Meaning</b>
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 ARMv8.2 and ARMv8.1:

Set Privileged Access Never, on taking an exception to EL1.

<b>SPAN</b>	<b>Meaning</b>
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 ARMv8.2:

Implicit Error Synchronizaition Barrier enable. Permitted values are:

<b>IESB</b>	<b>Meaning</b>
0	Disabled.
1	An implicit ErrorSynchronizationBarrier() call is added: <ul style="list-style-type: none"> <li>After each exception taken to EL1.</li> <li>Before the operational pseudocode of each ERET instruction executed at EL1.</li> </ul>

When the PE is in Debug state, the effect of this field is CONSTRAINED UNPREDICTABLE, and regardless of the value of the field its effective value might be 0 or 1. If the effective value of the field is 1, then an implicit ErrorSynchronizationBarrier() is added after each DCPSx instruction and before each DRPS instruction, in addition to the other cases where it is added.

This field is part of the required ARMv8.2 implementation of the RAS Extension. See 'The Reliability, Availability, and Serviceability (RAS) Extension' in the ARMv8 ARM, chapter A1 'Introduction to the ARMv8 Architecture'.

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 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.

**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 <a href="#">CTR_EL0</a> 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 <a href="#">DC ZVA</a> instruction at EL0 using AArch64 is trapped to EL1. Reading <a href="#">DCZID_EL0</a> .DZP from EL0 returns 1, indicating that <a href="#">DC ZVA</a> 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.

### Bit [13]

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 <a href="#">SCTLR_EL1</a> .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 <a href="#">SCTLR_EL1</a> .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 <a href="#">DAIF</a> 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"> <li>All encodings of the IT instruction with <code>hw1[3:0] != 1000</code>.</li> <li>All encodings of the subsequent instruction with the following values for <code>hw1</code>: <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> </li> </ul> <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"> <li>A 16-bit instruction, that can only be followed by another 16-bit instruction.</li> <li>The first half of a 32-bit instruction.</li> </ul> <p>This means that, for the situations that are UNDEFINED, either the second 16-bit instruction or the 32-bit instruction is UNDEFINED. 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 <a href="#">CP15DMB</a> , <a href="#">CP15DSB</a> , and <a href="#">CP15ISB</a> instructions is UNDEFINED.
1	EL0 using AArch32: EL0 execution of the <a href="#">CP15DMB</a> , <a href="#">CP15DSB</a> , and <a href="#">CP15ISB</a> 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"> <li>Data access to Normal memory from EL0 and EL1.</li> <li>Normal memory accesses to the EL1&amp;0 stage 1 translation tables.</li> </ul>

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	<a href="#">SCTLR_EL2</a>	RW
SCTLR_EL1	1	1	1	-	n/a	<a href="#">SCTLR_EL2</a>	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.

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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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
0	0	1	1	0	0	EE	0	1	1	0	0	WXN	1	0	1	0	0	0	I	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.

### Bits [31:30]

Reserved, RES0.

### Bits [29:28]

Reserved, RES1.

### Bits [27:26]

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.

#### Bits [21:20]

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 [15:13]

Reserved, RES0.

**I, bit [12]**

Instruction access Cacheability control, for accesses at EL2:

<b>I</b>	<b>Meaning</b>
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.

<b>C</b>	<b>Meaning</b>
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"> <li>• Data access to Normal memory from EL2.</li> <li>• Normal memory accesses to the EL2 translation tables.</li> </ul>

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}:

3130	29	28	27	26	25	24	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	LSMAOE	nTLSMD	0	UCI	EE	E0E	SPAN	1	IESB	1	WXN	nTWE	0	nTWI	UCT	DZE	0	1	1	0	0	SED	ITD	0	CP15	BEN	SA0	SAC	AM

This format applies only from ARMv8.1 and only in Non-secure state when [HCR\\_EL2](#).{E2H, TGE} == {1, 1}.

### Bits [31:30]

Reserved, RES0.

### LSMAOE, bit [29]

In 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.

In ARMv8.1:

Reserved, RES1.

### nTLSMD, bit [28]

In 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.

#### In ARMv8.1:

Reserved, RES1.

#### Bit [27]

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 <a href="#">DC CVAU</a> , <a href="#">DC CIVAC</a> , <a href="#">DC CVAC</a> , or <a href="#">IC IVAU</a> instruction at EL0 using AArch64 is 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.

#### 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 ARMv8.2:

Implicit Error Synchronizaiton Barrier enable. Permitted values are:

IESB	Meaning
0	Disabled.
1	An implicit ErrorSynchronizationBarrier() call is added: <ul style="list-style-type: none"> <li>After each exception taken to EL2.</li> <li>Before the operational pseudocode of each ERET instruction executed at EL2.</li> </ul>

When the PE is in Debug state, the effect of this field is CONSTRAINED UNPREDICTABLE, and regardless of the value of the field its effective value might be 0 or 1. If the effective value of the field is 1, then an implicit ErrorSynchronizationBarrier() is added after each DCPSx instruction and before each DRPS instruction, in addition to the other cases where it is added.

This field is part of the required ARMv8.2 implementation of the RAS Extension. See 'The Reliability, Availability, and Serviceability (RAS) Extension' in the ARMv8 ARM, chapter A1 'Introduction to the ARMv8 Architecture'.

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 <a href="#">CTR_EL0</a> 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 <a href="#">DC ZVA</a> instruction at EL0 using AArch64 is trapped to EL2. Reading <a href="#">DCZID_EL0.DZP</a> from EL0 returns 1, indicating that <a href="#">DC ZVA</a> 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.

#### Bit [13]

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"> <li>All encodings of the IT instruction with <code>hw1[3:0] != 1000</code>.</li> <li>All encodings of the subsequent instruction with the following values for <code>hw1</code>: <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> </li> </ul> <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"> <li>A 16-bit instruction, that can only be followed by another 16-bit instruction.</li> <li>The first half of a 32-bit instruction.</li> </ul> <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 <a href="#">CP15DMB</a> , <a href="#">CP15DSB</a> , and <a href="#">CP15ISB</a> instructions is UNDEFINED.
1	EL0 using AArch32: EL0 execution of the <a href="#">CP15DMB</a> , <a href="#">CP15DSB</a> , and <a href="#">CP15ISB</a> 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"> <li>Data access to Normal memory from EL2 and EL0.</li> <li>Normal memory accesses to the EL2&amp;0 translation tables.</li> </ul>

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	-	<a href="#">SCTLR_EL1</a>	n/a	<a href="#">SCTLR_EL1</a>
SCTLR_EL1	0	0	1	-	<a href="#">SCTLR_EL1</a>	<a href="#">SCTLR_EL1</a>	<a href="#">SCTLR_EL1</a>
SCTLR_EL1	0	1	1	-	n/a	<a href="#">SCTLR_EL1</a>	<a href="#">SCTLR_EL1</a>
SCTLR_EL1	1	0	1	-	<a href="#">SCTLR_EL1</a>	RW	<a href="#">SCTLR_EL1</a>
SCTLR_EL1	1	1	1	-	n/a	RW	<a href="#">SCTLR_EL1</a>

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.

# 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
0	0	1	1	0	0	EE	0	1	1	IESB	0	WXN	1	0	1	0	0	0	I	1	0	0	0	0	0	1	1	SA	C	A	M

### Bits [31:30]

Reserved, RES0.

### Bits [29:28]

Reserved, RES1.

### Bits [27:26]

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 ARMv8.2:**

Implicit Error Synchronizaition Barrier enable. Permitted values are:

IESB	Meaning
0	Disabled.
1	An implicit ErrorSynchronizationBarrier() call is added: <ul style="list-style-type: none"> <li>After each exception taken to EL3.</li> <li>Before the operational pseudocode of each ERET instruction executed at EL3.</li> </ul>

When the PE is in Debug state, the effect of this field is CONSTRAINED UNPREDICTABLE, and regardless of the value of the field its effective value might be 0 or 1. If the effective value of the field is 1, then an implicit ErrorSynchronizationBarrier() is added after each DCPSx instruction and before each DRPS instruction, in addition to the other cases where it is added.

This field is part of the required ARMv8.2 implementation of the RAS Extension. See 'The Reliability, Availability, and Serviceability (RAS) Extension' in the ARMv8 ARM, chapter A1 'Introduction to the ARMv8 Architecture'.

**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 [15:13]**

Reserved, RES0.

**I, bit [12]**

Instruction access Cacheability control, for accesses at EL3:

<b>I</b>	<b>Meaning</b>
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.

<b>C</b>	<b>Meaning</b>
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"> <li>Data access to Normal memory from EL3.</li> <li>Normal memory accesses to the EL3 translation tables.</li> </ul>

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.

# SDER32\_EL3, AArch32 Secure Debug Enable Register

The SDER32\_EL3 characteristics are:

## Purpose

Allows access to the AArch32 register [SDER](#) from AArch64 state only. Its value has no effect on execution in AArch64 state.

This register is part of:

- The Debug registers functional group.
- The Security registers functional group.

## Configuration

AArch64 System register SDER32\_EL3 is architecturally mapped to AArch32 System register [SDER](#).

If EL1 is AArch64 only, this register is UNDEFINED.

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

SDER32\_EL3 is a 32-bit register.

## Field descriptions

The SDER32\_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
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	<a href="#">SUNIDEN</a>	<a href="#">SUIDEN</a>

### Bits [31:2]

Reserved, RES0.

### SUNIDEN, bit [1]

Secure User Non-Invasive Debug Enable:

SUNIDEN	Meaning
0	Performance Monitors event counting prohibited in Secure EL0 unless allowed by <a href="#">MDCR_EL3</a> .SPME or the IMPLEMENTATION DEFINED authentication interface ExternalSecureNoninvasiveDebugEnabled().
1	Performance Monitors event counting allowed in Secure EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### SUIDEN, bit [0]

Secure User Invasive Debug Enable:

SUIDEN	Meaning
0	Debug exceptions other than Breakpoint Instruction exceptions from Secure EL0 are disabled, unless enabled by <a href="#">MDCR_EL3</a> .SPD32.
1	Debug exceptions from Secure EL0 are enabled.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the SDER32\_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
SDER_EL3	11	110	0001	0001	001

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# SPSR\_EL1, Saved Program Status Register (EL1)

The SPSR\_EL1 characteristics are:

## Purpose

Holds the saved process state when an exception is taken to EL1.

This register is part of the Special-purpose registers functional group.

## Configuration

AArch64 System register SPSR\_EL1 is architecturally mapped to AArch32 System register [SPSR\\_svc](#).

## Attributes

SPSR\_EL1 is a 32-bit register.

## Field descriptions

The SPSR\_EL1 bit assignments are:

### When exception taken from AArch32:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	Z	C	V	Q	IT[1:0]	J	0	PAN	SS	IL			GE					IT[7:2]				E	A	I	F	T	M[4]		M[3:0]		

An exception return from EL1 using AArch64 makes SPSR\_EL1 become UNKNOWN.

#### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to Supervisor mode, and copied to [CPSR.N](#) on executing an exception return operation in Supervisor mode.

#### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to Supervisor mode, and copied to [CPSR.Z](#) on executing an exception return operation in Supervisor mode.

#### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to Supervisor mode, and copied to [CPSR.C](#) on executing an exception return operation in Supervisor mode.

#### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to Supervisor mode, and copied to [CPSR.V](#) on executing an exception return operation in Supervisor mode.

#### Q, bit [27]

Cumulative saturation bit. Set to 1 to indicate that overflow or saturation occurred in some instructions.

**IT[1:0], bits [26:25]**

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]**

**In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR.PAN](#) on taking an exception to Supervisor mode, and copied to [CPSR.PAN](#) on executing an exception return operation in Supervisor mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**SS, bit [21]**

Software step. Shows the value of PSTATE.SS immediately before the exception was taken.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTL.R.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

SError interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

IRQ mask bit. The possible values of this bit are:

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

FIQ mask bit. The possible values of this bit are:

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0111	Abort
0b1011	Undefined
0b1111	System

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.

**When exception taken from AArch64:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
N	Z	C	V	0	0	0	0	U	A	O	P	A	N	S	S	I	L	0	0	0	0	0	0	0	0	D	A	I	F	0	M[4]	M[3:0]

An exception return from EL1 using AArch64 makes SPSR\_EL1 become UNKNOWN.

**N, bit [31]**

Set to the value of the N condition flag on taking an exception to EL1, and copied to the N condition flag on executing an exception return operation in EL1.

**Z, bit [30]**

Set to the value of the Z condition flag on taking an exception to EL1, and copied to the Z condition flag on executing an exception return operation in EL1.

**C, bit [29]**

Set to the value of the C condition flag on taking an exception to EL1, and copied to the C condition flag on executing an exception return operation in EL1.

**V, bit [28]**

Set to the value of the V condition flag on taking an exception to EL1, and copied to the V condition flag on executing an exception return operation in EL1.

**Bits [27:24]**

Reserved, RES0.

**UAO, bit [23]****In ARMv8.2:**

When ARMv8.2-UAO is implemented, set to the value of PSTATE.UAO on taking an exception to EL1, and copied to PSTATE.UAO on executing an exception return operation in EL1.

When ARMv8.2-UAO is not implemented, this bit is RES0.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of PSTATE.PAN on taking an exception to EL1, and copied to PSTATE.PAN on executing an exception return operation in EL1.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**SS, bit [21]**

Software step. Shows the value of PSTATE.SS immediately before the exception was taken.



**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**Bits [19:10]**

Reserved, RES0.

**D, bit [9]**

Process state D mask. The possible values of this bit are:

<b>D</b>	<b>Meaning</b>
0	Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are not masked.
1	Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are masked.

When the target Exception level of the debug exception is higher than the current Exception level, the exception is not masked by this bit.

**A, bit [8]**

SError interrupt mask bit. The possible values of this bit are:

<b>A</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**I, bit [7]**

IRQ mask bit. The possible values of this bit are:

<b>I</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**F, bit [6]**

FIQ mask bit. The possible values of this bit are:

<b>F</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**Bit [5]**

Reserved, RES0.

**M[4], bit [4]**

Execution state that the exception was taken from. Possible values of this bit are:

<b>M[4]</b>	<b>Meaning</b>
0	Exception taken from AArch64.

**M[3:0], bits [3:0]**

AArch64 state (Exception level and selected SP) that an exception was taken from. The possible values are:

M[3:0]	State
0b0000	EL0t
0b0100	EL1t
0b0101	EL1h

Other values are reserved, and returning to an Exception level that is using AArch64 with a reserved value in this field is treated as an illegal exception return.

The bits in this field are interpreted as follows:

- M[3:2] holds the Exception Level.
- M[1] is unused and is RES0 for all non-reserved values.
- M[0] is used to select the SP:
  - 0 means the SP is always SP0.
  - 1 means the exception SP is determined by the EL.

## Accessing the SPSR\_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
SPSR_EL1	11	000	0100	0000	000
SPSR_EL12	11	101	0100	0000	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
SPSR_EL1	x	x	0	-	RW	n/a	RW
SPSR_EL1	0	0	1	-	RW	RW	RW
SPSR_EL1	0	1	1	-	n/a	RW	RW
SPSR_EL1	1	0	1	-	RW	<a href="#">SPSR_EL2</a>	RW
SPSR_EL1	1	1	1	-	n/a	<a href="#">SPSR_EL2</a>	RW
SPSR_EL12	x	x	0	-	-	n/a	-
SPSR_EL12	0	0	1	-	-	-	-
SPSR_EL12	0	1	1	-	n/a	-	-
SPSR_EL12	1	0	1	-	-	RW	RW
SPSR_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic SPSR\_EL1 or SPSR\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

# SPSR\_EL2, Saved Program Status Register (EL2)

The SPSR\_EL2 characteristics are:

## Purpose

Holds the saved process state when an exception is taken to EL2.

This register is part of the Special-purpose registers functional group.

## Configuration

AArch64 System register SPSR\_EL2 is architecturally mapped to AArch32 System register [SPSR\\_hyp](#).

## Attributes

SPSR\_EL2 is a 32-bit register.

## Field descriptions

The SPSR\_EL2 bit assignments are:

### When exception taken from AArch32:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	Z	C	V	Q	IT[1:0]	J	0	PAN	SS	IL	GE			IT[7:2]			E			A	I	F	T	M[4]	M[3:0]						

An exception return from EL2 using AArch64 makes SPSR\_EL2 become UNKNOWN.

#### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to Hyp mode, and copied to [CPSR.N](#) on executing an exception return operation in Hyp mode.

#### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to Hyp mode, and copied to [CPSR.Z](#) on executing an exception return operation in Hyp mode.

#### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to Hyp mode, and copied to [CPSR.C](#) on executing an exception return operation in Hyp mode.

#### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to Hyp mode, and copied to [CPSR.V](#) on executing an exception return operation in Hyp mode.

#### Q, bit [27]

Cumulative saturation bit. Set to 1 to indicate that overflow or saturation occurred in some instructions.

#### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR](#).PAN on taking an exception to Hyp mode, and copied to [CPSR](#).PAN on executing an exception return operation in Hyp mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**SS, bit [21]**

Software step. Shows the value of PSTATE.SS immediately before the exception was taken.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTL.R.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

IRQ mask bit. The possible values of this bit are:

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

FIQ mask bit. The possible values of this bit are:

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0111	Abort
0b1010	Hyp
0b1011	Undefined
0b1111	System

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.

**When exception taken from AArch64:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	Z	C	V	0	0	0	0	UAO	PAN	SS	IL	0	0	0	0	0	0	0	0	0	0	D	A	I	F	0	M[4]		M[3:0]		

An exception return from EL2 using AArch64 makes SPSR\_EL2 become UNKNOWN.

**N, bit [31]**

Set to the value of the N condition flag on taking an exception to EL2, and copied to the N condition flag on executing an exception return operation in EL2.

**Z, bit [30]**

Set to the value of the Z condition flag on taking an exception to EL2, and copied to the Z condition flag on executing an exception return operation in EL2.

**C, bit [29]**

Set to the value of the C condition flag on taking an exception to EL2, and copied to the C condition flag on executing an exception return operation in EL2.

**V, bit [28]**

Set to the value of the V condition flag on taking an exception to EL2, and copied to the V condition flag on executing an exception return operation in EL2.

**Bits [27:24]**

Reserved, RES0.

**UAO, bit [23]****In ARMv8.2:**

When ARMv8.2-UAO is implemented, set to the value of PSTATE.UAO on taking an exception to EL2, and copied to PSTATE.UAO on executing an exception return operation in EL2.

When ARMv8.2-UAO is not implemented, this bit is RES0.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of PSTATE.PAN on taking an exception to EL2, and copied to PSTATE.PAN on executing an exception return operation in EL2.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**SS, bit [21]**

Software step. Shows the value of PSTATE.SS immediately before the exception was taken.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**Bits [19:10]**

Reserved, RES0.

**D, bit [9]**

Process state D mask. The possible values of this bit are:

<b>D</b>	<b>Meaning</b>
0	Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are not masked.
1	Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are masked.

When the target Exception level of the debug exception is higher than the current Exception level, the exception is not masked by this bit.

**A, bit [8]**

SError interrupt mask bit. The possible values of this bit are:

<b>A</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**I, bit [7]**

IRQ mask bit. The possible values of this bit are:

<b>I</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**F, bit [6]**

FIQ mask bit. The possible values of this bit are:

<b>F</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**Bit [5]**

Reserved, RES0.

**M[4], bit [4]**

Execution state that the exception was taken from. Possible values of this bit are:

<b>M[4]</b>	<b>Meaning</b>
0	Exception taken from AArch64.

**M[3:0], bits [3:0]**

AArch64 state (Exception level and selected SP) that an exception was taken from. The possible values are:

M[3:0]	State
0b0000	EL0t
0b0100	EL1t
0b0101	EL1h
0b1000	EL2t
0b1001	EL2h

Other values are reserved, and returning to an Exception level that is using AArch64 with a reserved value in this field is treated as an illegal exception return.

The bits in this field are interpreted as follows:

- M[3:2] holds the Exception Level.
- M[1] is unused and is RES0 for all non-reserved values.
- M[0] is used to select the SP:
  - 0 means the SP is always SP0.
  - 1 means the exception SP is determined by the EL.

## Accessing the SPSR\_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
SPSR_EL2	11	100	0100	0000	000
SPSR_EL1	11	000	0100	0000	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
SPSR_EL2	x	x	0	-	-	n/a	RW
SPSR_EL2	0	0	1	-	-	RW	RW
SPSR_EL2	0	1	1	-	n/a	RW	RW
SPSR_EL2	1	0	1	-	-	RW	RW
SPSR_EL2	1	1	1	-	n/a	RW	RW
SPSR_EL1	x	x	0	-	<a href="#">SPSR_EL1</a>	n/a	<a href="#">SPSR_EL1</a>
SPSR_EL1	0	0	1	-	<a href="#">SPSR_EL1</a>	<a href="#">SPSR_EL1</a>	<a href="#">SPSR_EL1</a>
SPSR_EL1	0	1	1	-	n/a	<a href="#">SPSR_EL1</a>	<a href="#">SPSR_EL1</a>
SPSR_EL1	1	0	1	-	<a href="#">SPSR_EL1</a>	RW	<a href="#">SPSR_EL1</a>
SPSR_EL1	1	1	1	-	n/a	RW	<a href="#">SPSR_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic SPSR\_EL2 or SPSR\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.



# SPSR\_EL3, Saved Program Status Register (EL3)

The SPSR\_EL3 characteristics are:

## Purpose

Holds the saved process state when an exception is taken to EL3.

This register is part of the Special-purpose registers functional group.

## Configuration

AArch64 System register SPSR\_EL3 can be mapped to AArch32 System register [SPSR\\_mon](#), but this is not architecturally mandated.

## Attributes

SPSR\_EL3 is a 32-bit register.

## Field descriptions

The SPSR\_EL3 bit assignments are:

### When exception taken from AArch32:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	Z	C	V	Q	IT[1:0]	J	0	PAN	SS	IL	GE			IT[7:2]			E			A	I	F	T	M[4]	M[3:0]						

An exception return from EL3 using AArch64 makes SPSR\_EL3 become UNKNOWN.

#### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to Monitor mode, and copied to [CPSR.N](#) on executing an exception return operation in Monitor mode.

#### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to Monitor mode, and copied to [CPSR.Z](#) on executing an exception return operation in Monitor mode.

#### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to Monitor mode, and copied to [CPSR.C](#) on executing an exception return operation in Monitor mode.

#### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to Monitor mode, and copied to [CPSR.V](#) on executing an exception return operation in Monitor mode.

#### Q, bit [27]

Cumulative saturation bit. Set to 1 to indicate that overflow or saturation occurred in some instructions.

**IT[1:0], bits [26:25]**

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]**

**In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR.PAN](#) on taking an exception to Monitor mode, and copied to [CPSR.PAN](#) on executing an exception return operation in Monitor mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**SS, bit [21]**

Software step. Shows the value of PSTATE.SS immediately before the exception was taken.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTL.R.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

SError interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

IRQ mask bit. The possible values of this bit are:

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

FIQ mask bit. The possible values of this bit are:

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor
0b0111	Abort
0b1010	Hyp
0b1011	Undefined
0b1111	System

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.

## When exception taken from AArch64:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
N	Z	C	V	0	0	0	0	UAO	PAN	SS	IL	0	0	0	0	0	0	0	0	0	0	D	A	I	F	0	M[4]		M[3:0]		

An exception return from EL3 using AArch64 makes SPSR\_EL3 become UNKNOWN.

### N, bit [31]

Set to the value of the N condition flag on taking an exception to EL3, and copied to the N condition flag on executing an exception return operation in EL3.

### Z, bit [30]

Set to the value of the Z condition flag on taking an exception to EL3, and copied to the Z condition flag on executing an exception return operation in EL3.

### C, bit [29]

Set to the value of the C condition flag on taking an exception to EL3, and copied to the C condition flag on executing an exception return operation in EL3.

### V, bit [28]

Set to the value of the V condition flag on taking an exception to EL3, and copied to the V condition flag on executing an exception return operation in EL3.

### Bits [27:24]

Reserved, RES0.

### UAO, bit [23]

In ARMv8.2:

When ARMv8.2-UAO is implemented, set to the value of PSTATE.UAO on taking an exception to EL3, and copied to PSTATE.UAO on executing an exception return operation in EL3.

When ARMv8.2-UAO is not implemented, this bit is RES0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

### PAN, bit [22]

In ARMv8.2 and ARMv8.1:

When ARMv8.1-PAN is implemented, set to the value of PSTATE.PAN on taking an exception to EL3, and copied to PSTATE.PAN on executing an exception return operation in EL3.

When ARMv8.1-PAN is not implemented, this bit is RES0.

In ARMv8.0:

Reserved, RES0.

**SS, bit [21]**

Software step. Shows the value of PSTATE.SS immediately before the exception was taken.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**Bits [19:10]**

Reserved, RES0.

**D, bit [9]**

Process state D mask. The possible values of this bit are:

<b>D</b>	<b>Meaning</b>
0	Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are not masked.
1	Watchpoint, Breakpoint, and Software Step exceptions targeted at the current Exception level are masked.

When the target Exception level of the debug exception is higher than the current Exception level, the exception is not masked by this bit.

**A, bit [8]**

SError interrupt mask bit. The possible values of this bit are:

<b>A</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**I, bit [7]**

IRQ mask bit. The possible values of this bit are:

<b>I</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**F, bit [6]**

FIQ mask bit. The possible values of this bit are:

<b>F</b>	<b>Meaning</b>
0	Exception not masked.
1	Exception masked.

**Bit [5]**

Reserved, RES0.

**M[4], bit [4]**

Execution state that the exception was taken from. Possible values of this bit are:

<b>M[4]</b>	<b>Meaning</b>
0	Exception taken from AArch64.

**M[3:0], bits [3:0]**

AArch64 state (Exception level and selected SP) that an exception was taken from. The possible values are:

<b>M[3:0]</b>	<b>State</b>
0b0000	EL0t
0b0100	EL1t
0b0101	EL1h
0b1000	EL2t
0b1001	EL2h
0b1100	EL3t
0b1101	EL3h

Other values are reserved, and returning to an Exception level that is using AArch64 with a reserved value in this field is treated as an illegal exception return.

The bits in this field are interpreted as follows:

- M[3:2] holds the Exception Level.
- M[1] is unused and is RES0 for all non-reserved values.
- M[0] is used to select the SP:
  - 0 means the SP is always SP0.
  - 1 means the exception SP is determined by the EL.

## Accessing the SPSR\_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:

<b>&lt;systemreg&gt;</b>	<b>op0</b>	<b>op1</b>	<b>CRn</b>	<b>CRm</b>	<b>op2</b>
SPSR_EL3	11	110	0100	0000	000

## Accessibility

The register is accessible as follows:

<b>Control</b>			<b>Accessibility</b>			
<b>E2H</b>	<b>TGE</b>	<b>NS</b>	<b>EL0</b>	<b>EL1</b>	<b>EL2</b>	<b>EL3</b>
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.

# SPSR\_abt, Saved Program Status Register (Abort mode)

The SPSR\_abt characteristics are:

## Purpose

Holds the saved process state when an exception is taken to Abort mode.

This register is part of the Special-purpose registers functional group.

## Configuration

AArch64 System register SPSR\_abt is architecturally mapped to AArch32 System register [SPSR\\_abt](#).

If EL1 does not support execution in AArch32 state, this register is RES0.

## Attributes

SPSR\_abt is a 32-bit register.

## Field descriptions

The SPSR\_abt 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
N	Z	C	V	Q	IT[1:0]	J	0	PAN	0	IL	GE		IT[7:2]				E		A	I	F	T	M[4]	M[3:0]							

### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to Abort mode, and copied to [CPSR.N](#) on executing an exception return operation in Abort mode.

### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to Abort mode, and copied to [CPSR.Z](#) on executing an exception return operation in Abort mode.

### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to Abort mode, and copied to [CPSR.C](#) on executing an exception return operation in Abort mode.

### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to Abort mode, and copied to [CPSR.V](#) on executing an exception return operation in Abort mode.

### Q, bit [27]

Cumulative saturation bit. Set to 1 to indicate that overflow or saturation occurred in some instructions.

### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See [IT\[7:2\]](#) for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR](#).PAN on taking an exception to Abort mode, and copied to [CPSR](#).PAN on executing an exception return operation in Abort mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**Bit [21]**

Reserved, RES0.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTL.R.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.



If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

IRQ mask bit. The possible values of this bit are:

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

FIQ mask bit. The possible values of this bit are:

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor
0b0111	Abort
0b1010	Hyp
0b1011	Undefined
0b1111	System

Other values are reserved.

## Accessing the SPSR\_abt

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
SPSR_abt	11	100	0100	0011	001

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# SPSR\_fiq, Saved Program Status Register (FIQ mode)

The SPSR\_fiq characteristics are:

## Purpose

Holds the saved process state when an exception is taken to FIQ mode.

This register is part of the Special-purpose registers functional group.

## Configuration

AArch64 System register SPSR\_fiq is architecturally mapped to AArch32 System register [SPSR\\_fiq](#).

If EL1 does not support execution in AArch32 state, this register is RES0.

## Attributes

SPSR\_fiq is a 32-bit register.

## Field descriptions

The SPSR\_fiq 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
N	Z	C	V	Q	IT[1:0]	J	0	PAN	0	IL	GE				IT[7:2]						E	A	I	F	T	M[4]	M[3:0]				

### N, bit [31]

Set to the value of [CPSR](#).N on taking an exception to FIQ mode, and copied to [CPSR](#).N on executing an exception return operation in FIQ mode.

### Z, bit [30]

Set to the value of [CPSR](#).Z on taking an exception to FIQ mode, and copied to [CPSR](#).Z on executing an exception return operation in FIQ mode.

### C, bit [29]

Set to the value of [CPSR](#).C on taking an exception to FIQ mode, and copied to [CPSR](#).C on executing an exception return operation in FIQ mode.

### V, bit [28]

Set to the value of [CPSR](#).V on taking an exception to FIQ mode, and copied to [CPSR](#).V on executing an exception return operation in FIQ mode.

### Q, bit [27]

Cumulative saturation bit. Set to 1 to indicate that overflow or saturation occurred in some instructions.

### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

### J, bit [24]

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]**

In ARMv8.2 and ARMv8.1:

When ARMv8.1-PAN is implemented, set to the value of CPSR.PAN on taking an exception to FIQ mode, and copied to CPSR.PAN on executing an exception return operation in FIQ mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

In ARMv8.0:

Reserved, RES0.

**Bit [21]**

Reserved, RES0.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTL.R.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

**A, bit [8]**

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

**I, bit [7]**

IRQ mask bit. The possible values of this bit are:

I	Meaning
0	Exception not masked.
1	Exception masked.

**F, bit [6]**

FIQ mask bit. The possible values of this bit are:

F	Meaning
0	Exception not masked.
1	Exception masked.

**T, bit [5]**

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

**M[4], bit [4]**

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

**M[3:0], bits [3:0]**

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor
0b0111	Abort
0b1010	Hyp
0b1011	Undefined
0b1111	System

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.

## Accessing the SPSR\_fiq

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
SPSR_fiq	11	100	0100	0011	011

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# SPSR\_irq, Saved Program Status Register (IRQ mode)

The SPSR\_irq characteristics are:

## Purpose

Holds the saved process state when an exception is taken to IRQ mode.  
This register is part of the Special-purpose registers functional group.

## Configuration

AArch64 System register SPSR\_irq is architecturally mapped to AArch32 System register [SPSR\\_irq](#).  
If EL1 does not support execution in AArch32 state, this register is RES0.

## Attributes

SPSR\_irq is a 32-bit register.

## Field descriptions

The SPSR\_irq 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
N	Z	C	V	Q	IT[1:0]	J	0	PAN	0	IL	GE			IT[7:2]					E	A	I	F	T	M[4]	M[3:0]						

### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to IRQ mode, and copied to [CPSR.N](#) on executing an exception return operation in IRQ mode.

### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to IRQ mode, and copied to [CPSR.Z](#) on executing an exception return operation in IRQ mode.

### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to IRQ mode, and copied to [CPSR.C](#) on executing an exception return operation in IRQ mode.

### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to IRQ mode, and copied to [CPSR.V](#) on executing an exception return operation in IRQ mode.

### Q, bit [27]

Cumulative saturation bit. Set to 1 to indicate that overflow or saturation occurred in some instructions.

### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR](#).PAN on taking an exception to IRQ mode, and copied to [CPSR](#).PAN on executing an exception return operation in IRQ mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**Bit [21]**

Reserved, RES0.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTLR.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.



If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

IRQ mask bit. The possible values of this bit are:

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

FIQ mask bit. The possible values of this bit are:

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor
0b0111	Abort
0b1010	Hyp
0b1011	Undefined
0b1111	System

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.

## Accessing the SPSR\_irq

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
SPSR_irq	11	100	0100	0011	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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# SPSR\_und, Saved Program Status Register (Undefined mode)

The SPSR\_und characteristics are:

## Purpose

Holds the saved process state when an exception is taken to Undefined mode.  
This register is part of the Special-purpose registers functional group.

## Configuration

AArch64 System register SPSR\_und is architecturally mapped to AArch32 System register [SPSR\\_und](#).  
If EL1 does not support execution in AArch32 state, this register is RES0.

## Attributes

SPSR\_und is a 32-bit register.

## Field descriptions

The SPSR\_und 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
<a href="#">N</a>	<a href="#">Z</a>	<a href="#">C</a>	<a href="#">V</a>	<a href="#">Q</a>	<a href="#">IT[1:0]</a>	<a href="#">J</a>	0	<a href="#">PAN</a>	0	<a href="#">IL</a>	<a href="#">GE</a>		<a href="#">IT[7:2]</a>				<a href="#">E</a>		<a href="#">A</a>	<a href="#">I</a>	<a href="#">F</a>	<a href="#">T</a>	<a href="#">M[4]</a>	<a href="#">M[3:0]</a>							

### N, bit [31]

Set to the value of [CPSR.N](#) on taking an exception to Undefined mode, and copied to [CPSR.N](#) on executing an exception return operation in Undefined mode.

### Z, bit [30]

Set to the value of [CPSR.Z](#) on taking an exception to Undefined mode, and copied to [CPSR.Z](#) on executing an exception return operation in Undefined mode.

### C, bit [29]

Set to the value of [CPSR.C](#) on taking an exception to Undefined mode, and copied to [CPSR.C](#) on executing an exception return operation in Undefined mode.

### V, bit [28]

Set to the value of [CPSR.V](#) on taking an exception to Undefined mode, and copied to [CPSR.V](#) on executing an exception return operation in Undefined mode.

### Q, bit [27]

Cumulative saturation bit. Set to 1 to indicate that overflow or saturation occurred in some instructions.

### IT[1:0], bits [26:25]

IT block state bits for the T32 IT (If-Then) instruction. See IT[7:2] for explanation of this field.

**J, bit [24]**

RES0.

In previous versions of the architecture, the {J, T} bits determined the AArch32 Instruction set state. ARMv8 does not support either Jazelle state or T32EE state, and the T bit determines the Instruction set state.

**Bit [23]**

Reserved, RES0.

**PAN, bit [22]****In ARMv8.2 and ARMv8.1:**

When ARMv8.1-PAN is implemented, set to the value of [CPSR.PAN](#) on taking an exception to Undefined mode, and copied to [CPSR.PAN](#) on executing an exception return operation in Undefined mode.

When ARMv8.1-PAN is not implemented, this bit is RES0.

**In ARMv8.0:**

Reserved, RES0.

**Bit [21]**

Reserved, RES0.

**IL, bit [20]**

Illegal Execution state bit. Shows the value of PSTATE.IL immediately before the exception was taken.

**GE, bits [19:16]**

Greater than or Equal flags, for parallel addition and subtraction.

**IT[7:2], bits [15:10]**

IT block state bits for the T32 IT (If-Then) instruction. This field must be interpreted in two parts.

- IT[7:5] holds the base condition for the IT block. The base condition is the top 3 bits of the condition code specified by the first condition field of the IT instruction.
- IT[4:0] encodes the size of the IT block, which is the number of instructions that are to be conditionally executed, by the position of the least significant 1 in this field. It also encodes the value of the least significant bit of the condition code for each instruction in the block.

The IT field is 0b00000000 when no IT block is active.

**E, bit [9]**

Endianness state bit. Controls the load and store endianness for data accesses:

E	Meaning
0	Little-endian operation
1	Big-endian operation.

Instruction fetches ignore this bit.

When the reset value of the SCTL.R.EE bit is defined by a configuration input signal, that value also applies to the CPSR.E bit on reset, and therefore applies to software execution from reset.

If an implementation does not provide Big-endian support, this bit is RES0. If it does not provide Little-endian support, this bit is RES1.

If an implementation provides Big-endian support but only at EL0, this bit is RES0 for an exception return to any Exception level other than EL0.

Likewise, if it provides Little-endian support only at EL0, this bit is RES1 for an exception return to any Exception level other than EL0.

#### A, bit [8]

Error interrupt mask bit. The possible values of this bit are:

A	Meaning
0	Exception not masked.
1	Exception masked.

#### I, bit [7]

IRQ mask bit. The possible values of this bit are:

I	Meaning
0	Exception not masked.
1	Exception masked.

#### F, bit [6]

FIQ mask bit. The possible values of this bit are:

F	Meaning
0	Exception not masked.
1	Exception masked.

#### T, bit [5]

T32 Instruction set state bit. Determines the AArch32 instruction set state that the exception was taken from. Possible values of this bit are:

T	Meaning
0	Taken from A32 state.
1	Taken from T32 state.

#### M[4], bit [4]

Execution state that the exception was taken from. Possible values of this bit are:

M[4]	Meaning
1	Exception taken from AArch32.

#### M[3:0], bits [3:0]

AArch32 mode that an exception was taken from. The possible values are:

M[3:0]	Mode
0b0000	User
0b0001	FIQ
0b0010	IRQ
0b0011	Supervisor
0b0110	Monitor
0b0111	Abort
0b1010	Hyp
0b1011	Undefined
0b1111	System

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.

## Accessing the SPSR\_und

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
SPSR_und	11	100	0100	0011	010

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# SPSel, Stack Pointer Select

The SPSel characteristics are:

## Purpose

Allows the Stack Pointer to be selected between SP\_EL0 and SP\_ELx.

This register is part of the Process state registers functional group.

## Configuration

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 AArch64. Otherwise, RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

SPSel is a 32-bit register.

## Field descriptions

The SPSel 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	SP

### Bits [31:1]

Reserved, RES0.

### SP, bit [0]

Stack pointer to use. Possible values of this bit are:

SP	Meaning
0	Use SP_EL0 at all Exception levels.
1	Use SP_ELx for Exception level ELx.

When this register has an architecturally-defined reset value, this field resets to 1.

## Accessing the SPSel

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
SPSel	11	000	0100	0010	000

This register can be modified using MSR (immediate) with the following syntax:

MSR &lt;pstatefield&gt;, &lt;imm&gt;

This syntax uses the following encoding in the System instruction encoding space:

<pstatefield>	op0	op1	CRn	op2
SPSel	00	000	0100	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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# SP\_EL0, Stack Pointer (EL0)

The SP\_EL0 characteristics are:

## Purpose

Holds the stack pointer associated with EL0. At higher Exception levels, this is used as the current stack pointer when the value of [SPSel.SP](#) is 0.

This register is part of the Special-purpose registers functional group.

## Configuration

There are no configuration notes.

## Attributes

SP\_EL0 is a 64-bit register.

## Field descriptions

The SP\_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
Stack pointer																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Stack pointer																															

### Bits [63:0]

Stack pointer.

## Accessing the SP\_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
SP_EL0	11	000	0100	0001	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.

This accessibility information only applies when the value of [SPSel.SP](#) is 1, and only for accesses using the MRS or MSR instructions. In addition, this register is accessible at EL0 as the current stack pointer.

When the value of [SPSel.SP](#) is 0:

- Any access to SP\_EL0 using the MRS or MSR instructions is UNDEFINED.
- This register is accessible at all Exception levels as the current stack pointer.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# SP\_EL1, Stack Pointer (EL1)

The SP\_EL1 characteristics are:

## Purpose

Holds the stack pointer associated with EL1. When executing at EL1, the value of [SPSel.SP](#) determines the current stack pointer:

SPSel.SP	current stack pointer
0	<a href="#">SP_EL0</a>
1	<a href="#">SP_EL1</a>

This register is part of the Special-purpose registers functional group.

## Configuration

There are no configuration notes.

## Attributes

SP\_EL1 is a 64-bit register.

## Field descriptions

The SP\_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
Stack pointer																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Stack pointer																															

### Bits [63:0]

Stack pointer.

## Accessing the SP\_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
SP_EL1	11	100	0100	0001	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.

This accessibility information only applies to accesses using the MRS or MSR instructions.

When the value of [SPSel.SP](#) is 1, this register is also accessible at EL1 as the current stack pointer.

---

**Note**

When the value of [SPSel.SP](#) is 0, [SP\\_EL0](#) is used as the current stack pointer at all Exception levels.

---

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# SP\_EL2, Stack Pointer (EL2)

The SP\_EL2 characteristics are:

## Purpose

Holds the stack pointer associated with EL2. When executing at EL2, the value of [SPSel.SP](#) determines the current stack pointer:

SPSel.SP	current stack pointer
0	<a href="#">SP_EL0</a>
1	<a href="#">SP_EL2</a>

This register is part of the Special-purpose registers functional group.

## Configuration

There are no configuration notes.

## Attributes

SP\_EL2 is a 64-bit register.

## Field descriptions

The SP\_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
																Stack pointer															
																Stack pointer															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Stack pointer.

## Accessing the SP\_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
SP_EL2	11	110	0100	0001	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.

This accessibility information only applies to accesses using the MRS or MSR instructions.

When the value of [SPSel.SP](#) is 1, this register is also accessible at EL2 as the current stack pointer.

---

**Note**

When the value of [SPSel.SP](#) is 0, [SP\\_EL0](#) is used as the current stack pointer at all Exception levels.

---

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# SP\_EL3, Stack Pointer (EL3)

The SP\_EL3 characteristics are:

## Purpose

Holds the stack pointer associated with EL3. When executing at EL3, the value of [SPSel.SP](#) determines the current stack pointer:

SPSel.SP	current stack pointer
0	<a href="#">SP_EL0</a>
1	<a href="#">SP_EL3</a>

This register is part of the Special-purpose registers functional group.

## Usage constraints

This register is not accessible using MRS and MSR instructions.

When the value of [SPSel.SP](#) is 1, this register is accessible at EL3 as the current stack pointer.

---

### Note

When the value of [SPSel.SP](#) is 0, [SP\\_EL0](#) is used as the current stack pointer at all Exception levels.

---

## Traps and Enables

There are no traps or enables affecting this register.

## Configuration

There are no configuration notes.

## Attributes

SP\_EL3 is a 64-bit register.

## Field descriptions

The SP\_EL3 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						
														Stack pointer																							
														Stack pointer																							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						

### Bits [63:0]

Stack pointer.

## 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	42	41		
0	0	0	0	0	0	0	0	0	NFD1	NFD0	0	0	HWU162	HWU161	HWU160	HWU159	HWU062	HWU061	HWU060	HWU059	HPD1	HPD0		
TG1	SH1	ORGN1	IRGN1	EPD1	A1				T1SZ						TG0			SH0			ORGN0			IRGN0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9		

Any of the bits in TCR\_EL1 are permitted to be cached in a TLB.

**Bits [63:55]**

Reserved, RES0.

**NFD1, bit [54]**

**In ARMv8.2:**

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 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:

<b>NFD1</b>	<b>Meaning</b>
0	Perform translation table walks using <a href="#">TTBR1_EL1</a> .
1	A TLB miss on an address that is translated using <a href="#">TTBR1_EL1</a> 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 ARMv8.2:**

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 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 <a href="#">TTBR0_EL1</a> .
1	A TLB miss on an address that is translated using <a href="#">TTBR0_EL1</a> 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.

**Bits [52:51]**

Reserved, RES0.

**HWU162, bit [50]**

**In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR1\\_EL1](#) if the TCR\_EL1.HPD1 value is 1.

Defined values are:

HWU162	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL1.HPD1 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU161, bit [49]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR1\\_EL1](#) if the TCR\_EL1.HPD1 value is 1.

Defined values are:

HWU161	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL1.HPD1 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU160, bit [48]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR1\\_EL1](#) if the TCR\_EL1.HPD1 value is 1.

Defined values are:

HWU160	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL1.HPD1 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU159, bit [47]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR1\\_EL1](#) if the TCR\_EL1.HPD1 value is 1.

Defined values are:

HWU159	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL1.HPD1 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU062, bit [46]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR0\\_EL1](#) if the TCR\_EL1.HPD0 value is 1.

Defined values are:

HWU062	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL1.HPD0 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU061, bit [45]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR0\\_EL1](#) if the TCR\_EL1.HPD0 value is 1.

Defined values are:

HWU061	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL1.HPD0 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU060, bit [44]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR0\\_EL1](#) if the TCR\_EL1.HPD0 value is 1.

Defined values are:

HWU060	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL1.HPD0 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU059, bit [43]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR0\\_EL1](#) if the TCR\_EL1.HPD0 value is 1.

Defined values are:

HWU059	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL1.HPD0 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HPD1, bit [42]****In 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 ARMv8.2 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 ARMv8.2 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 ARMv8.2 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.

Additionally, this affects changes to the program counter, when TBI1 is 1 and bit [55] of the target address is 1, caused by:

- A branch or procedure return within EL0 or EL1.
- An exception taken to EL1.
- An exception return to EL0 or EL1.

In these cases bits [63:56] of the address are also set to 1 before it is stored in the PC.

**TBI0, bit [37]**

Top Byte ignored - indicates whether the top byte of an address is used for address match for the [TTBR0\\_EL1](#) region, or ignored and used for tagged addresses. 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 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.

Additionally, this affects changes to the program counter, when TBI0 is 1 and bit [55] of the target address is 0, caused by:

- A branch or procedure return within EL0 or EL1.
- An exception taken to EL1.
- An exception return to EL0 or EL1.

In these cases bits [63:56] of the address are also set to 0 before it is stored in the PC.

## AS, bit [36]

ASID Size. Defined values are:

AS	Meaning
0	8 bit - the upper 8 bits of <a href="#">TTBR0_EL1</a> and <a href="#">TTBR1_EL1</a> 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 <a href="#">TTBR0_EL1</a> and <a href="#">TTBR1_EL1</a> 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 <a href="#">TTBR1_EL1</a> .
1	A TLB miss on an address that is translated using <a href="#">TTBR1_EL1</a> 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	<a href="#">TTBR0_EL1</a> .ASID defines the ASID.
1	<a href="#">TTBR1_EL1</a> .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 <a href="#">TTBR0_EL1</a> .
1	A TLB miss on an address that is translated using <a href="#">TTBR0_EL1</a> 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	<a href="#">TCR_EL2</a>	RW
TCR_EL1	1	1	1	-	n/a	<a href="#">TCR_EL2</a>	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.

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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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	0	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.

### Bits [30:29]

Reserved, RES0.

**HWU62, bit [28]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table block or level 3 entry if the TCR\_EL2.HPD value is 1.

Defined values are:

HWU62	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL2.HPD value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU61, bit [27]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table block or level 3 entry if the TCR\_EL2.HPD value is 1.

Defined values are:

HWU61	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL2.HPD value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU60, bit [26]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table block or level 3 entry if the TCR\_EL2.HPD value is 1.

Defined values are:

HWU60	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL2.HPD bit value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU59, bit [25]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table block or level 3 entry if the TCR\_EL2.HPD value is 1.

Defined values are:

HWU59	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL2.HPD bit value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HPD, bit [24]****In 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 [TTBRO\\_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 ARMv8.2 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 ARMv8.2 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]

Additionally, this affects changes to the program counter, when TBI is 1, caused by:

- A branch or procedure return within EL2.
- An exception taken to EL2.
- An exception return to EL2.

In these cases bits [63:56] of the address are set to 0 before it is stored in the PC.

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.

#### 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	42	41
0	0	0	0	0	0	0	0	0	NFD1	NFD0	0	0	HWU162	HWU161	HWU160	HWU159	HWU062	HWU061	HWU060	HWU059	HPD1	HPD0
TG1	SH1	ORGN1	IRGN1	EPD1	A1												TG0			SH0		ORGN0
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9

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 ARMv8.2:**

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 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 <a href="#">TTBR1_EL2</a> .
1	A TLB miss on an address that is translated using <a href="#">TTBR1_EL2</a> 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 ARMv8.2:**

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 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 <a href="#">TTBR0_EL2</a> .
1	A TLB miss on an address that is translated using <a href="#">TTBR0_EL2</a> 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.

#### Bits [52:51]

Reserved, RES0.

#### HWU162, bit [50]

##### In ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR1\\_EL2](#) if the TCR\_EL2.HPD1 value is 1.

Defined values are:

HWU162	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL2.HPD1 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

#### In ARMv8.1:

Reserved, RES0.

#### HWU161, bit [49]

##### In ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR1\\_EL2](#) if the TCR\_EL2.HPD1 value is 1.

Defined values are:

HWU161	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL2.HPD1 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

#### In ARMv8.1:

Reserved, RES0.

**HWU160, bit [48]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR1\\_EL2](#) if the TCR\_EL2.HPD1 value is 1.

Defined values are:

HWU160	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL2.HPD1 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1:**

Reserved, RES0.

**HWU159, bit [47]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR1\\_EL2](#) if the TCR\_EL2.HPD1 value is 1.

Defined values are:

HWU159	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL2.HPD1 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1:**

Reserved, RES0.

**HWU062, bit [46]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR0\\_EL2](#) if the TCR\_EL2.HPD0 value is 1.

Defined values are:

HWU062	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL2.HPD0 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1:**

Reserved, RES0.

**HWU061, bit [45]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR0\\_EL2](#) if the TCR\_EL2.HPD0 value is 1.

Defined values are:

HWU061	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL2.HPD0 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1:**

Reserved, RES0.

**HWU060, bit [44]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR0\\_EL2](#) if the TCR\_EL2.HPD0 value is 1.

Defined values are:

HWU060	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL2.HPD0 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1:**

Reserved, RES0.

**HWU059, bit [43]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table block or level 3 entry for pages pointed to by [TTBR0\\_EL2](#) if the TCR\_EL2.HPD0 value is 1.

Defined values are:

HWU059	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL2.HPD0 value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1:**

Reserved, RES0.

**HPD1, bit [42]**

Hierarchical Permission Disables. This affects the hierarchical control bits, APTable, 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, APTable, 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]**

Additionally, this affects changes to the program counter, when TBI1 is 1 and bit [55] of the target address is 1, caused by:

- A branch or procedure return within EL0 or EL1.
- An exception taken to EL1.
- An exception return to EL0 or EL1.

In these cases bits [63:56] of the address are also set to 1 before it is stored in the PC.

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.

### TBIO, bit [37]

Additionally, this affects changes to the program counter, when TBIO is 1 and bit [55] of the target address is 0, caused by:

- A branch or procedure return within EL0 or EL1.
- An exception taken to EL1.
- An exception return to EL0 or EL1.

In these cases bits [63:56] of the address are also set to 0 before it is stored in the PC.

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.

### AS, bit [36]

ASID Size. Defined values are:

AS	Meaning
0	8 bit - the upper 8 bits of <a href="#">TTBR0_EL2</a> and <a href="#">TTBR1_EL2</a> 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 <a href="#">TTBR0_EL2</a> and <a href="#">TTBR1_EL2</a> 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 <a href="#">TTBR1_EL2</a> .
1	A TLB miss on an address that is translated using <a href="#">TTBR1_EL2</a> 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	<a href="#">TTBR0_EL2</a> .ASID defines the ASID.
1	<a href="#">TTBR1_EL2</a> .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 <a href="#">TTBR0_EL2</a> .
1	A TLB miss on an address that is translated using <a href="#">TTBR0_EL2</a> 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	-	<a href="#">TCR_EL1</a>	n/a	<a href="#">TCR_EL1</a>
TCR_EL1	0	0	1	-	<a href="#">TCR_EL1</a>	<a href="#">TCR_EL1</a>	<a href="#">TCR_EL1</a>
TCR_EL1	0	1	1	-	n/a	<a href="#">TCR_EL1</a>	<a href="#">TCR_EL1</a>
TCR_EL1	1	0	1	-	<a href="#">TCR_EL1</a>	RW	<a href="#">TCR_EL1</a>
TCR_EL1	1	1	1	-	n/a	RW	<a href="#">TCR_EL1</a>

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.



# TCR\_EL3, Translation Control Register (EL3)

The TCR\_EL3 characteristics are:

## Purpose

The control register for stage 1 of the EL3 translation regime.

This register is part of the Virtual memory control registers functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

TCR\_EL3 is a 32-bit register.

## Field descriptions

The TCR\_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
1	0	0	HWU62	HWU61	HWU60	HWU59	HPD	1	HD	HA	TBI	0	PS	TG0	SH0	ORGN0	IRGN0	0	0												T0SZ

Any of the bits in TCR\_EL3 are permitted to be cached in a TLB.

### Bit [31]

Reserved, RES1.

### Bits [30:29]

Reserved, RES0.

### HWU62, bit [28]

In ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 1 translation table block or level 3 entry if the TCR\_EL3.HPD value is 1.

Defined values are:

HWU62	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL3.HPD value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

**HWU61, bit [27]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 1 translation table block or level 3 entry if the TCR\_EL3.HPD value is 1.

Defined values are:

HWU61	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL3.HPD bit value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU60, bit [26]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 1 translation table block or level 3 entry if the TCR\_EL3.HPD value is 1.

Defined values are:

HWU60	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL3.HPD bit value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU59, bit [25]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 1 translation table block or level 3 entry if the TCR\_EL3.HPD value is 1.

Defined values are:

HWU59	Meaning
0	The stage 1 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 1 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose if the TCR_EL3.HPD bit value is 1.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HPD, bit [24]****In 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\\_EL3](#).

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 ARMv8.2 and ARMv8.1:**

Hardware management of dirty state in stage 1 translations from EL3.

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 ARMv8.2 and ARMv8.1:**

Hardware Access flag update in stage 1 translations from EL3.

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\\_EL3](#) 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 EL3 using AArch64 where the address would be translated by tables pointed to by [TTBR0\\_EL3](#). It has an effect whether the EL3 translation regime is enabled or not.

Additionally, this affects changes to the program counter, when TBI is 1, caused by:

- A branch or procedure return within EL3.
- A exception taken to EL3.
- An exception return to EL3.

In these cases bits [63:56] of the address are set to 0 before it is stored in the PC.

**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\_EL3 are 0000.

**TG0, bits [15:14]**

Granule size for the [TTBR0\\_EL3](#).

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\\_EL3](#).

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\\_EL3](#).

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\\_EL3](#).

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\\_EL3](#). 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\_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
TCR_EL3	11	110	0010	0000	010

## 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
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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TPIDRRO\_EL0, EL0 Read-Only Software Thread ID Register

The TPIDRRO\_EL0 characteristics are:

## Purpose

Provides a location where software executing at EL1 or higher can store thread identifying information that is visible to software executing at EL0, for OS management purposes.

The PE makes no use of this register.

This register is part of the Thread and process ID registers functional group.

## Configuration

AArch64 System register TPIDRRO\_EL0 bits [31:0] are architecturally mapped to AArch32 System register [TPIDRURO](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

TPIDRRO\_EL0 is a 64-bit register.

## Field descriptions

The TPIDRRO\_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
Thread ID																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

## Accessing the TPIDRRO\_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
TPIDRRO_EL0	11	011	1101	0000	011

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# TPIDR\_EL0, EL0 Read/Write Software Thread ID Register

The TPIDR\_EL0 characteristics are:

## Purpose

Provides a location where software executing at EL0 can store thread identifying information, for OS management purposes.

The PE makes no use of this register.

This register is part of the Thread and process ID registers functional group.

## Configuration

AArch64 System register TPIDR\_EL0 bits [31:0] are architecturally mapped to AArch32 System register [TPIDRURW](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

TPIDR\_EL0 is a 64-bit register.

## Field descriptions

The TPIDR\_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
Thread ID																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Thread ID																															

### Bits [63:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

## Accessing the TPIDR\_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
TPIDR_EL0	11	011	1101	0000	010

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TPIDR\_EL1, EL1 Software Thread ID Register

The TPIDR\_EL1 characteristics are:

## Purpose

Provides a location where software executing at EL1 can store thread identifying information, for OS management purposes.

The PE makes no use of this register.

This register is part of the Thread and process ID registers functional group.

## Configuration

AArch64 System register TPIDR\_EL1 bits [31:0] are architecturally mapped to AArch32 System register [TPIDRPRW](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

TPIDR\_EL1 is a 64-bit register.

## Field descriptions

The TPIDR\_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
Thread ID																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Thread ID																															

### Bits [63:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

## Accessing the TPIDR\_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
TPIDR_EL1	11	000	1101	0000	100

## Accessibility

The register is accessible as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3

# TPIDR\_EL1, EL1 Software Thread ID Register

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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TPIDR\_EL2, EL2 Software Thread ID Register

The TPIDR\_EL2 characteristics are:

## Purpose

Provides a location where software executing at EL2 can store thread identifying information, for OS management purposes.

The PE makes no use of this register.

This register is part of:

- The Virtualization registers functional group.
- The Thread and process ID registers functional group.

## Configuration

AArch64 System register TPIDR\_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HTPIDR](#).

If EL2 is not implemented, this register is RES0 from EL3.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

TPIDR\_EL2 is a 64-bit register.

## Field descriptions

The TPIDR\_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
Thread ID																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Thread ID																															

### Bits [63:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

## Accessing the TPIDR\_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
TPIDR_EL2	11	100	1101	0000	010

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TPIDR\_EL3, EL3 Software Thread ID Register

The TPIDR\_EL3 characteristics are:

## Purpose

Provides a location where software executing at EL3 can store thread identifying information, for OS management purposes.

The PE makes no use of this register.

This register is part of the Thread and process ID registers functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

TPIDR\_EL3 is a 64-bit register.

## Field descriptions

The TPIDR\_EL3 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
Thread ID																															
Thread ID																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Thread ID. Thread identifying information stored by software running at this Exception level.

## Accessing the TPIDR\_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
TPIDR_EL3	11	110	1101	0000	010

## 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

# TPIDR\_EL3, EL3 Software Thread ID Register

x	1	1	-	n/a	-	RW
---	---	---	---	-----	---	----

This table applies to all instructions that can access this register.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# TTBR0\_EL1, Translation Table Base Register 0 (EL1)

The TTBR0\_EL1 characteristics are:

## Purpose

Holds the base address of the translation table for the initial lookup for stage 1 of the translation of an address from the lower VA range in the EL1&0 translation regime, and other information for this translation regime.

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch64 System register TTBR0\_EL1 is architecturally mapped to AArch32 System register [TTBR0](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

TTBR0\_EL1 is a 64-bit register.

## Field descriptions

The TTBR0\_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	
ASID																BADDR																
BADDR																															CnF	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Any of the fields in this register are permitted to be cached in a TLB.

### ASID, bits [63:48]

An ASID for the translation table base address. The [TCR\\_EL1.A1](#) field selects either TTBR0\_EL1.ASID or TTBR1\_EL1.ASID.

If the implementation has only 8 bits of ASID, then the upper 8 bits of this field are RES0.

### BADDR, bits [47:1]

Translation table base address, A[47:x] or A[51:x], bits[47:1].

#### Note

- Translation table base addresses of 52 bits, A[51:x], are supported only in an implementation that includes ARMv8.2-LPA and is using the 64KB translation granule.
- A translation table must be aligned to the size of the table, except that when using a translation table base address larger than 48 bits the minimum alignment of a table containing fewer than eight entries is 64 bytes.

In an implementation that includes ARMv8.2-LPA, if the value of [TCR\\_EL1.IPS](#) is 110, then:

- Register bits[47:z] hold bits[47:z] of the stage 1 translation table base address, where z is determined as follows:
  - If  $x \geq 6$  then  $z=x$ .
  - Otherwise,  $z=6$ .
- Register bits[5:2] hold bits[51:48] of the stage 1 translation table base address.
- When  $z > x$  register bits[(z-1):x] are RES0, and bits[(z-1):x] of the translation table base address are zero.
- When  $x > 6$  register bits[(x-1):6] are RES0.
- Register bit[1] is RES0.
- Bits[5:2] of the stage 1 translation table base address are zero.

- In an implementation that includes ARMv8.2-TTCNP bit[0] of the stage 1 translation table base address is zero.

---

**Note**

In an implementation that includes ARMv8.2-LPA a [TCR\\_EL1](#).IPS value of 110, that selects an IPA size of 52 bits, is permitted only when using the 64KB translation granule.

---

When the value of [ID\\_AA64MMFR0\\_EL1](#).PARange indicates that the implementation does not support a 52 bit PA size, if a translation table lookup uses this register with the 64KB translation granule when the value of [TCR\\_EL1](#).IPS is 110 and the value of register bits[5:2] is nonzero it is IMPLEMENTATION DEFINED whether an Address size fault is generated, but ARM deprecates not generating an Address size fault.

If the Effective value of [TCR\\_EL1](#).IPS is not 110 then:

- Register bits[47:x] hold bits[47:x] of the stage 1 translation table base address.
  - Register bits[(x-1):1] are RES0.
  - If the implementation supports 52-bit PAs and IPAs then bits[51:49] of the translation table base addresses used in this stage of translation are 0b0000.
- 

**Note**

This definition applies:

- To an implementation that includes ARMv8.2-LPA and is using a translation granule smaller than 64KB.
  - To any implementation that does not include ARMv8.2-LPA.
- 

If any TTBR0\_EL1[47:0] bit that is defined as RES0 has the value 1 when a translation table walk is performed using TTBR0\_EL1, then the translation table base address might be misaligned, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Bits[x-1:0] of the translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [TCR\\_EL1](#).T0SZ, the stage of translation, and the translation granule size.

**CnP, bit [0]**

In ARMv8.2:

Common not Private. In an implementation that includes ARMv8.2-TTCNP, indicates whether each entry that is pointed to by TTBR0\_EL1 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR0\_EL1.CnP is 1.

CnP	Meaning
0	The translation table entries pointed to by TTBR0_EL1, for the current translation regime and ASID, are permitted to differ from corresponding entries for TTBR0_EL1 for other PEs in the Inner Shareable domain. This is not affected by: <ul style="list-style-type: none"> <li>• The value of TTBR0_EL1.CnP on those other PEs.</li> <li>• The value of the current ASID or, in Non-secure state, the value of the current VMID.</li> </ul>
1	The translation table entries pointed to by TTBR0_EL1 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR0_EL1.CnP is 1 and all of the following apply: <ul style="list-style-type: none"> <li>• The translation table entries are pointed to by TTBR0_EL1.</li> <li>• The translation tables relate to the same translation regime.</li> <li>• The ASID is the same as the current ASID.</li> <li>• In Non-secure state, the VMID is the same as the current VMID.</li> </ul>

---

When a TLB combines entries from stage 1 translation and stage 2 translation into a single entry, that entry can only be shared between different PEs if the value of the CnP bit is 1 for both stage 1 and stage 2.

---

**Note**

If the value of the TTBR0\_EL1.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR0\_EL1s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONSTRAINED UNPREDICTABLE, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values' in the ARMv8-A ARM appendix K1.

---

In an implementation that does not include ARMv8.2-TTCNP this field is RES0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

## Accessing the TTBR0\_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
TTBR0_EL1	11	000	0010	0000	000
TTBR0_EL12	11	101	0010	0000	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
TTBR0_EL1	x	x	0	-	RW	n/a	RW
TTBR0_EL1	0	0	1	-	RW	RW	RW
TTBR0_EL1	0	1	1	-	n/a	RW	RW
TTBR0_EL1	1	0	1	-	RW	<a href="#">TTBR0_EL2</a>	RW
TTBR0_EL1	1	1	1	-	n/a	<a href="#">TTBR0_EL2</a>	RW
TTBR0_EL12	x	x	0	-	-	n/a	-
TTBR0_EL12	0	0	1	-	-	-	-
TTBR0_EL12	0	1	1	-	n/a	-	-
TTBR0_EL12	1	0	1	-	-	RW	RW
TTBR0_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic TTBR0\_EL1 or TTBR0\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

## 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.

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.



# TTBR0\_EL2, Translation Table Base Register 0 (EL2)

The TTBR0\_EL2 characteristics are:

## Purpose

When [HCR\\_EL2.E2H](#) is 0, holds the base address of the translation table for the initial lookup for stage 1 of an address translation in the EL2 translation regime, and other information for this translation regime.

When [HCR\\_EL2.E2H](#) is 1, holds the base address of the translation table for the initial lookup for stage 1 of the translation of an address from the lower VA range in the EL2&0 translation regime, and other information for this translation regime.

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

## Configuration

AArch64 System register TTBR0\_EL2 is architecturally mapped to AArch32 System register [HTTBR](#).

If EL2 is not implemented, this register is RES0 from EL3.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

TTBR0\_EL2 is a 64-bit register.

## Field descriptions

The TTBR0\_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	
ASID																BADDR																
																BADDR																CnF
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

**Note**

- Translation table base addresses of 52 bits, A[51:x], are supported only in an implementation that includes ARMv8.2-LPA and is using the 64KB translation granule.
- A translation table must be aligned to the size of the table, except that when using a translation table base address larger than 48 bits the minimum alignment of a table containing fewer than eight entries is 64 bytes.

In an implementation that includes ARMv8.2-LPA, if the value of [TCR\\_EL2](#).{I}PS is 110, then:

- Register bits[47:z] hold bits[47:z] of the stage 1 translation table base address, where z is determined as follows:
  - If  $x \geq 6$  then  $z=x$ .
  - Otherwise,  $z=6$ .
- Register bits[5:2] hold bits[51:48] of the stage 1 translation table base address.
- When  $z > x$  register bits[(z-1):x] are RES0, and bits[(z-1):x] of the translation table base address are zero.
- When  $x > 6$  register bits[(x-1):6] are RES0.
- Register bit[1] is RES0.
- Bits[5:2] of the stage 1 translation table base address are zero.
- In an implementation that includes ARMv8.2-TTCNP bit[0] of the stage 1 translation table base address is zero.

**Note**

In an implementation that includes ARMv8.2-LPA:

- A [TCR\\_EL2](#).{I}PS value of 110, that selects an OA size of 52 bits, is permitted only when using the 64KB translation granule.
- The OA size is specified by:
  - The value of [TCR\\_EL2](#).PS when the value of [HCR\\_EL2](#).E2H is 0.
  - The value of [TCR\\_EL2](#).IPS when the value of [HCR\\_EL2](#).E2H is 1.

When the value of [ID\\_AA64MMFR0\\_EL1](#).PARange indicates that the implementation does not support a 52 bit PA size, if a translation table lookup uses this register with the 64KB translation granule when the value of [TCR\\_EL2](#).{I}PS is 110 and the value of register bits[5:2] is nonzero it is IMPLEMENTATION DEFINED whether an Address size fault is generated, but ARM deprecates not generating an Address size fault.

If the Effective value of [TCR\\_EL2](#).{I}PS is not 110 then:

- Register bits[47:x] hold bits[47:x] of the stage 1 translation table base address.
- Register bits[(x-1):1] are RES0.
- If the implementation supports 52-bit PAs and IPAs then bits[51:49] of the translation table base addresses used in this stage of translation are 0b0000.

**Note**

This definition applies:

- To an implementation that includes ARMv8.2-LPA and is using a translation granule smaller than 64KB.
- To any implementation that does not include ARMv8.2-LPA.

If any TTBR0\_EL2[47:0] bit that is defined as RES0 has the value 1 when a translation table walk is performed using TTBR0\_EL2, then the translation table base address might be misaligned, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Bits[x-1:0] of the translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [TCR\\_EL2](#).T0SZ, the stage of translation, and the translation granule size.

**CnP, bit [0]****In ARMv8.2:**

Common not Private. In an implementation that includes ARMv8.2-TTCNP, indicates whether each entry that is pointed to by TTBR0\_EL2 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR0\_EL2.CnP is 1.

CnP	Meaning
0	The translation table entries pointed to by TTBR0_EL2 for the current translation regime, and ASID if applicable, are permitted to differ from corresponding entries for TTBR0_EL2 for other PEs in the Inner Shareable domain. This is not affected by: <ul style="list-style-type: none"> <li>The value of TTBR0_EL2.CnP on those other PEs.</li> <li>When the current translation regime is the EL2&amp;0 regime, the value of the current ASID.</li> </ul>
1	The translation table entries pointed to by TTBR0_EL2 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR0_EL2.CnP is 1 and all of the following apply: <ul style="list-style-type: none"> <li>The translation table entries are pointed to by TTBR0_EL2.</li> <li>The translation tables relate to the same translation regime.</li> <li>If that translation regime is the EL2&amp;0 regime, the ASID is the same as the current ASID.</li> </ul>

#### Note

If the value of the TTBR0\_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR0\_EL2s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONSTRAINED UNPREDICTABLE, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values' in the ARMv8-A ARM appendix K1.

In an implementation that does not include ARMv8.2-TTCNP this field is RES0.

#### In ARMv8.1 and ARMv8.0:

Reserved, RES0.

## Accessing the TTBR0\_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
TTBR0_EL2	11	100	0010	0000	000
TTBR0_EL1	11	000	0010	0000	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
TTBR0_EL2	x	x	0	-	-	n/a	RW
TTBR0_EL2	0	0	1	-	-	RW	RW
TTBR0_EL2	0	1	1	-	n/a	RW	RW
TTBR0_EL2	1	0	1	-	-	RW	RW
TTBR0_EL2	1	1	1	-	n/a	RW	RW
TTBR0_EL1	x	x	0	-	<a href="#">TTBR0_EL1</a>	n/a	<a href="#">TTBR0_EL1</a>
TTBR0_EL1	0	0	1	-	<a href="#">TTBR0_EL1</a>	<a href="#">TTBR0_EL1</a>	<a href="#">TTBR0_EL1</a>
TTBR0_EL1	0	1	1	-	n/a	<a href="#">TTBR0_EL1</a>	<a href="#">TTBR0_EL1</a>

# TTBR0\_EL2, Translation Table Base Register 0 (EL2)

TTBR0_EL1	1	0	1	-	<a href="#">TTBR0_EL1</a>	RW	<a href="#">TTBR0_EL1</a>
TTBR0_EL1	1	1	1	-	n/a	RW	<a href="#">TTBR0_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic TTBR0\_EL2 or TTBR0\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# TTBR0\_EL3, Translation Table Base Register 0 (EL3)

The TTBR0\_EL3 characteristics are:

## Purpose

Holds the base address of the translation table for the initial lookup for stage 1 of an address translation in the EL3 translation regime, and other information for this translation regime.

This register is part of the Virtual memory control registers functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

TTBR0\_EL3 is a 64-bit register.

## Field descriptions

The TTBR0\_EL3 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	BADDR															
BADDR																CnP															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Any of the fields in this register are permitted to be cached in a TLB.

### Bits [63:48]

Reserved, RES0.

### BADDR, bits [47:1]

Translation table base address, A[47:x] or A[51:x].

#### Note

- Translation table base addresses of 52 bits, A[51:x], are supported only in an implementation that includes ARMv8.2-LPA and is using the 64KB translation granule.
- A translation table must be aligned to the size of the table, except that when using translation table base address larger than 48 bits the minimum alignment of a table containing fewer than eight entries is 64 bytes.

In an implementation that includes ARMv8.2-LPA, if the value of [TCR\\_EL3](#).PS is 110 then:

- Register bits[47:z] hold bits[47:z] of the stage 1 translation table base address, where z is determined as follows:
  - If  $x \geq 6$  then  $z=x$ .
  - Otherwise,  $z=6$ .
- Register bits[5:2] hold bits[51:48] of the stage 1 translation table base address.
- When  $z > x$  register bits[(z-1):x] are RES0, and bits[(z-1):x] of the translation table base address are zero.
- When  $x > 6$  register bits[(x-1):6] are RES0.
- Register bit[1] is RES0.
- Bits[5:2] of the stage 1 translation table base address are zero.
- In an implementation that includes ARMv8.2-TTCNP bit[0] of the stage 1 translation table base address is zero.

#### Note

In an implementation that includes ARMv8.2-LPA a [TCR\\_EL3](#).PS value of 110, that selects a PA size of 52 bits, is permitted only when using the 64KB translation granule.

When the value of [ID\\_AA64MMFR0\\_EL1](#).PARange indicates that the implementation does not support a 52 bit PA size, if a translation table lookup uses this register with the 64KB translation granule when the value of [TCR\\_EL3](#).PS is 110 and the value of register bits[5:2] is nonzero it is IMPLEMENTATION DEFINED whether an Address size fault is generated, but ARM deprecates not generating an Address size fault.

If the Effective value of [TCR\\_EL3](#).PS is not 110 then:

- Register bits[47:x] hold bits[47:x] of the stage 1 translation table base address.
- Register bits[(x-1):1] are RES0.
- If the implementation supports 52-bit PAs and IPAs then bits[51:49] of the translation table base addresses used in this stage of translation are 0b0000.

#### Note

This definition applies:

- To an implementation that includes ARMv8.2-LPA and is using a translation granule smaller than 64KB.
- To any implementation that does not include ARMv8.2-LPA.

If any TTBR0\_EL3[47:0] bit that is defined as RES0 has the value 1 when a translation table walk is performed using TTBR0\_EL3, then the translation table base address might be misaligned, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Bits[x-1:0] of the translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [TCR\\_EL3](#).TOSZ, the stage of translation, and the translation granule size.

#### CnP, bit [0]

In ARMv8.2:

Common not Private. In an implementation that includes ARMv8.2-TTCNP, indicates whether each entry that is pointed to by TTBR0\_EL3 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR0\_EL3.CnP is 1.

CnP	Meaning
0	The translation table entries pointed to by TTBR0_EL3, for the current translation regime, are permitted to differ from corresponding entries for TTBR0_EL3 for other PEs in the Inner Shareable domain. This is not affected by the value of TTBR0_EL3.CnP on those other PEs.
1	The translation table entries pointed to by TTBR0_EL3 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR0_EL3.CnP is 1 and the translation table entries are pointed to by TTBR0_EL3.

#### Note

If the value of the TTBR0\_EL3.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR0\_EL3s do not point to the same translation table entries the results of translations using TTBR0\_EL3 are CONSTRAINED UNPREDICTABLE, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values' in the ARMv8-A ARM appendix K1.

In an implementation that does not include ARMv8.2-TTCNP this field is RES0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

## Accessing the TTBR0\_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
TTBR0_EL3	11	110	0010	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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TTBR1\_EL1, Translation Table Base Register 1 (EL1)

The TTBR1\_EL1 characteristics are:

## Purpose

Holds the base address of the translation table for the initial lookup for stage 1 of the translation of an address from the higher VA range in the EL1&0 stage 1 translation regime, and other information for this translation regime.

This register is part of the Virtual memory control registers functional group.

## Configuration

AArch64 System register TTBR1\_EL1 is architecturally mapped to AArch32 System register [TTBR1](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

TTBR1\_EL1 is a 64-bit register.

## Field descriptions

The TTBR1\_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	
ASID																BADDR																
BADDR																CnP																
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

Any of the fields in this register are permitted to be cached in a TLB.

### ASID, bits [63:48]

An ASID for the translation table base address. The [TCR\\_EL1](#).A1 field selects either TTBR0\_EL1.ASID or TTBR1\_EL1.ASID.

If the implementation has only 8 bits of ASID, then the upper 8 bits of this field are RES0.

### BADDR, bits [47:1]

Translation table base address, A[47:x] or A[51:x], bits[47:1].

#### Note

- Translation table base addresses of 52 bits, A[51:x], are supported only in an implementation that includes ARMv8.2-LPA and is using the 64KB translation granule.
- A translation table must be aligned to the size of the table, except that when using a translation table base address larger than 48 bits the minimum alignment of a table containing fewer than eight entries is 64 bytes.

In an implementation that includes ARMv8.2-LPA, if the value of [TCR\\_EL1](#).IPS is 110, then:

- Register bits[47:z] hold bits[47:z] of the stage 1 translation table base address, where z is determined as follows:
  - If  $x \geq 6$  then  $z=x$ .
  - Otherwise,  $z=6$ .
- Register bits[5:2] hold bits[51:48] of the stage 1 translation table base address.
- When  $z > x$  register bits[(z-1):x] are RES0, and bits[(z-1):x] of the translation table base address are zero.
- When  $x > 6$  register bits[(x-1):6] are RES0.
- Register bit[1] is RES0.
- Bits[5:2] of the stage 1 translation table base address are zero.

- In an implementation that includes ARMv8.2-TTCNP bit[0] of the stage 1 translation table base address is zero.

---

**Note**

In an implementation that includes ARMv8.2-LPA a [TCR\\_EL1](#).IPS value of 110, that selects an IPA size of 52 bits, is permitted only when using the 64KB translation granule.

---

When the value of [ID\\_AA64MMFR0\\_EL1](#).PARange indicates that the implementation does not support a 52 bit PA size, if a translation table lookup uses this register with the 64KB translation granule when the value of [TCR\\_EL1](#).IPS is 110 and the value of register bits[5:2] is nonzero it is IMPLEMENTATION DEFINED whether an Address size fault is generated, but ARM deprecates not generating an Address size fault.

If the Effective value of [TCR\\_EL1](#).IPS is not 110 then:

- Register bits[47:x] hold bits[47:x] of the stage 1 translation table base address.
  - Register bits[(x-1):1] are RES0.
  - If the implementation supports 52-bit PAs and IPAs then bits[51:49] of the translation table base addresses used in this stage of translation are 0b0000.
- 

**Note**

This definition applies:

- To an implementation that includes ARMv8.2-LPA and is using a translation granule smaller than 64KB.
  - To any implementation that does not include ARMv8.2-LPA.
- 

If any TTBR1\_EL1[47:0] bit that is defined as RES0 has the value 1 when a translation table walk is performed using TTBR1\_EL1, then the translation table base address might be misaligned, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Bits[x-1:0] of the translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [TCR\\_EL1](#).T1SZ, the stage of translation, and the translation granule size.

**CnP, bit [0]**

In ARMv8.2:

Common not Private. In an implementation that includes ARMv8.2-TTCNP, indicates whether each entry that is pointed to by TBR1\_EL1 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR1\_EL1.CnP is 1.

CnP	Meaning
0	The translation table entries pointed to by TTBR1_EL1, for the current translation regime and ASID, are permitted to differ from corresponding entries for TTBR1_EL1 for other PEs in the Inner Shareable domain. This is not affected by: <ul style="list-style-type: none"> <li>• The value of TTBR1_EL1.CnP on those other PEs.</li> <li>• The value of the current ASID or, in Non-secure state, the value of the current VMID.</li> </ul>
1	The translation table entries pointed to by TTBR1_EL1 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR1_EL1.CnP is 1 and all of the following apply: <ul style="list-style-type: none"> <li>• The translation table entries are pointed to by TTBR1_EL1.</li> <li>• The translation tables relate to the same translation regime.</li> <li>• The ASID is the same as the current ASID.</li> <li>• In Non-secure state, the VMID is the same as the current VMID.</li> </ul>

---

When a TLB combines entries from stage 1 translation and stage 2 translation into a single entry, that entry can only be shared between different PEs if the value of the CnP bit is 1 for both stage 1 and stage 2.

---

**Note**

If the value of the TTBR1\_EL1.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR1\_EL1s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONSTRAINED UNPREDICTABLE, see 'CONSTRAINED UNPREDICTABLE behaviors due to caching of control or data values' in the ARMv8-A ARM appendix K1.

---

In an implementation that does not include ARMv8.2-TTCNP this field is RES0.

In ARMv8.1 and ARMv8.0:

Reserved, RES0.

## Accessing the TTBR1\_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
TTBR1_EL1	11	000	0010	0000	001
TTBR1_EL12	11	101	0010	0000	001

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
TTBR1_EL1	x	x	0	-	RW	n/a	RW
TTBR1_EL1	0	0	1	-	RW	RW	RW
TTBR1_EL1	0	1	1	-	n/a	RW	RW
TTBR1_EL1	1	0	1	-	RW	<a href="#">TTBR1_EL2</a>	RW
TTBR1_EL1	1	1	1	-	n/a	<a href="#">TTBR1_EL2</a>	RW
TTBR1_EL12	x	x	0	-	-	n/a	-
TTBR1_EL12	0	0	1	-	-	-	-
TTBR1_EL12	0	1	1	-	n/a	-	-
TTBR1_EL12	1	0	1	-	-	RW	RW
TTBR1_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic TTBR1\_EL1 or TTBR1\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

## 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.

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.



## TTBR1\_EL2, Translation Table Base Register 1 (EL2)

The TTBR1\_EL2 characteristics are:

## Purpose

When [HCR\\_EL2.E2H](#) is 1, holds the base address of the translation table for the initial lookup for stage 1 of the translation of an address from the higher VA range in the EL2&0 translation regime, and other information for this translation regime.

### Note

When [HCR\\_EL2.E2H](#) is 0, the contents of this register are ignored by the PE, except for a direct read or write of the register.

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

## Configuration

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

TTBR1\_EL2 is a 64-bit register.

## Field descriptions

The TTBR1 EL2 bit assignments are:

Diagram illustrating the 64-bit instruction format (RV64I) for the RISC-V architecture. The instruction is divided into three main fields:

- Opcode (7 bits):** Bits 63 to 57.
- Register Identifiers (15 bits):** Bits 56 to 40, including *rs1*, *rs2*, and *rd*.
- Immediate Value (15 bits):** Bits 39 to 25, including the sign-extended immediate (bits 39 to 26) and the zero-extension field (bits 25 to 23, labeled *CnP*).

Any of the fields in this register are permitted to be cached in a TLB.

**ASID, bits [63:48]**

An ASID for the translation table base address. The [TCR\\_EL2.A1](#) field selects either TTBR0\_EL2.ASID or TTBR1\_EL2.ASID.

If the implementation has only 8 bits of ASID, then the upper 8 bits of this field are RES0.

**BADDR, bits [47:1]**

Translation table base address, A[47:x] or A[51:x], bits[47:1].

### Note

- Translation table base addresses of 52 bits, A[51:x], are supported only in an implementation that includes ARMv8.2-LPA and is using the 64KB translation granule.
- A translation table must be aligned to the size of the table, except that when using a translation table base address larger than 48 bits the minimum alignment of a table containing fewer than eight entries is 64 bytes.



In an implementation that includes ARMv8.2-LPA, if the value of [TCR\\_EL2](#).{I}PS is 110, then:

- Register bits[47:z] hold bits[47:z] of the stage 1 translation table base address, where z is determined as follows:
  - If  $x \geq 6$  then  $z=x$ .
  - Otherwise,  $z=6$ .
- Register bits[5:2] hold bits[51:48] of the stage 1 translation table base address.
- When  $z > x$  register bits[(z-1):x] are RES0, and bits[(z-1):x] of the translation table base address are zero.
- When  $x > 6$  register bits[(x-1):6] are RES0.
- Register bit[1] is RES0.
- Bits[5:2] of the stage 1 translation table base address are zero.
- In an implementation that includes ARMv8.2-TTCNP bit[0] of the stage 1 translation table base address is zero.

#### Note

In an implementation that includes ARMv8.2-LPA a [TCR\\_EL2](#).IPS value of 110, that selects an OA size of 52 bits, is permitted only when using the 64KB translation granule.

When the value of [ID\\_AA64MMFR0\\_EL1](#).PARange indicates that the implementation does not support a 52 bit PA size, if a translation table lookup uses this register with the 64KB translation granule when the value of [TCR\\_EL2](#).IPS is 110 and the value of register bits[5:2] is nonzero it is IMPLEMENTATION DEFINED whether an Address size fault is generated, but ARM deprecates not generating an Address size fault.

If the Effective value of [TCR\\_EL2](#).IPS is not 110 then:

- Register bits[47:x] hold bits[47:x] of the stage 1 translation table base address.
- Register bits[(x-1):1] are RES0.
- If the implementation supports 52-bit PAs and IPAs then bits[51:49] of the translation table base addresses used in this stage of translation are 0b0000.

#### Note

This definition applies:

- To an implementation that includes ARMv8.2-LPA and is using a translation granule smaller than 64KB.
- To any implementation that does not include ARMv8.2-LPA.

If any TTBR1\_EL2[47:0] bit that is defined as RES0 has the value 1 when a translation table walk is performed using TTBR1\_EL2, then the translation table base address might be misaligned, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Bits[x-1:0] of the translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [TCR\\_EL2](#).T1SZ, the stage of translation, and the translation granule size.

### CnP, bit [0]

#### In ARMv8.2:

Common not Private. In an implementation that includes ARMv8.2-TTCNP, indicates whether each entry that is pointed to by TBR1\_EL2 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of TTBR1\_EL2.CnP is 1.

CnP	Meaning
0	The translation table entries pointed to by TTBR1_EL2 for the current ASID are permitted to differ from corresponding entries for TTBR1_EL2 for other PEs in the Inner Shareable domain. This is not affected by: <ul style="list-style-type: none"> <li>• The value of TTBR1_EL2.CnP on those other PEs.</li> <li>• The value of the current ASID.</li> </ul>
1	The translation table entries pointed to by TTBR1_EL2 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of TTBR1_EL2.CnP is 1 and all of the following apply: <ul style="list-style-type: none"> <li>• The translation table entries are pointed to by TTBR1_EL2.</li> <li>• The ASID is the same as the current ASID.</li> </ul>

#### Note

TTBR1\_EL2 is accessible only when the value of [HCR\\_EL2](#).E2H is 1, meaning the current translation regime is the EL2&0 regime.

If the value of the TTBR1\_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those TTBR1\_EL2s do not point to the same translation table entries when the other conditions specified for the case when the value of CnP is 1 apply, then the results of translations are CONstrained UNpredictable, see 'CONstrained UNpredictable behaviors due to caching of control or data values' in the ARMv8-A ARM appendix K1.

In an implementation that does not include ARMv8.2-TTCNP this field is RES0.

#### In ARMv8.1:

Reserved, RES0.

## Accessing the TTBR1\_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
TTBR1_EL2	11	100	0010	0000	001
TTBR1_EL1	11	000	0010	0000	001

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
TTBR1_EL2	x	x	0	-	-	n/a	RW
TTBR1_EL2	0	0	1	-	-	RW	RW
TTBR1_EL2	0	1	1	-	n/a	RW	RW
TTBR1_EL2	1	0	1	-	-	RW	RW
TTBR1_EL2	1	1	1	-	n/a	RW	RW
TTBR1_EL1	x	x	0	-	<a href="#">TTBR1_EL1</a>	n/a	<a href="#">TTBR1_EL1</a>
TTBR1_EL1	0	0	1	-	<a href="#">TTBR1_EL1</a>	<a href="#">TTBR1_EL1</a>	<a href="#">TTBR1_EL1</a>
TTBR1_EL1	0	1	1	-	n/a	<a href="#">TTBR1_EL1</a>	<a href="#">TTBR1_EL1</a>
TTBR1_EL1	1	0	1	-	<a href="#">TTBR1_EL1</a>	RW	<a href="#">TTBR1_EL1</a>
TTBR1_EL1	1	1	1	-	n/a	RW	<a href="#">TTBR1_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic TTBR1\_EL2 or TTBR1\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

# UAO, User Access Override

The UAO characteristics are:

## Purpose

When ARMv8.2-UAO is implemented, allows access to the User Access Override bit.

When ARMv8.2-UAO is not implemented, this register is not implemented.

This register is part of the Special-purpose registers functional group.

## Configuration

This register is introduced in ARMv8.2.

## Attributes

UAO is a 32-bit register.

## Field descriptions

The UAO 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	UAO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

### Bits [31:24]

Reserved, RES0.

### UAO, bit [23]

User Access Override. Defined values are:

UAO	Meaning
0	The behaviour of LDTR* and STTR* instructions is as defined in the base ARMv8 architecture.
1	When executed at EL1, or at EL2 with <a href="#">HCR_EL2</a> .{E2H, TGE} == {1, 1}, LDTR* and STTR* instructions behave as the equivalent LDR* and STR* instructions.

When executed at EL3, or at EL2 with [HCR\\_EL2](#).E2H == 0 or [HCR\\_EL2](#).TGE == 0, the LDTR\* and STTR\* instructions behave as the equivalent LDR\* and STR\* instructions, regardless of the setting of the PSTATE.UAO bit.

### Bits [22:0]

Reserved, RES0.

## Accessing the UAO

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
UAO	11	000	0100	0010	100

This register can be modified using MSR (immediate) with the following syntax:

```
MSR <pstatefield>, <imm>
```

This syntax uses the following encoding in the System instruction encoding space:

<pstatefield>	op0	op1	CRn	op2
UAO	00	000	0100	011

## 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.

# VBAR\_EL1, Vector Base Address Register (EL1)

The VBAR\_EL1 characteristics are:

## Purpose

Holds the vector base address for any exception that is taken to EL1.

This register is part of the Exception and fault handling registers functional group.

## Configuration

AArch64 System register VBAR\_EL1 bits [31:0] are architecturally mapped to AArch32 System register [VBAR](#).

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

VBAR\_EL1 is a 64-bit register.

## Field descriptions

The VBAR\_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
Vector Base Address																															
Vector Base Address																				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 [63:11]

Vector Base Address. Base address of the exception vectors for exceptions taken to EL1.

If the implementation does not support ARMv8.2-LVA, then:

- If tagged addresses are being used, bits [55:48] of VBAR\_EL1 must be the same or else the use of the vector address will result in a recursive exception.
- If tagged addresses are not being used, bits [63:48] of VBAR\_EL1 must be the same or else the use of the vector address will result in a recursive exception.

If the implementation supports ARMv8.2-LVA, then:

- If tagged addresses are being used, bits [55:52] of VBAR\_EL1 must be the same or else the use of the vector address will result in a recursive exception.
- If tagged addresses are not being used, bits [63:52] of VBAR\_EL1 must be the same or else the use of the vector address will result in a recursive exception.

### Bits [10:0]

Reserved, RES0.

## Accessing the VBAR\_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 &lt;systemreg&gt;, &lt;Xt&gt;

This syntax uses the following encoding in the System instruction encoding space:

<systemreg>	op0	op1	CRn	CRm	op2
VBAR_EL1	11	000	1100	0000	000
VBAR_EL12	11	101	1100	0000	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
VBAR_EL1	x	x	0	-	RW	n/a	RW
VBAR_EL1	0	0	1	-	RW	RW	RW
VBAR_EL1	0	1	1	-	n/a	RW	RW
VBAR_EL1	1	0	1	-	RW	<a href="#">VBAR_EL2</a>	RW
VBAR_EL1	1	1	1	-	n/a	<a href="#">VBAR_EL2</a>	RW
VBAR_EL12	x	x	0	-	-	n/a	-
VBAR_EL12	0	0	1	-	-	-	-
VBAR_EL12	0	1	1	-	n/a	-	-
VBAR_EL12	1	0	1	-	-	RW	RW
VBAR_EL12	1	1	1	-	n/a	RW	RW

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL3 using the mnemonic VBAR\_EL1 or VBAR\_EL12 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

# VBAR\_EL2, Vector Base Address Register (EL2)

The VBAR\_EL2 characteristics are:

## Purpose

Holds the vector base address for any exception that is taken to EL2.

This register is part of:

- The Virtualization registers functional group.
- The Exception and fault handling registers functional group.

## Configuration

AArch64 System register VBAR\_EL2 bits [31:0] are architecturally mapped to AArch32 System register [HVBAR](#).

If EL2 is not implemented, this register is RES0 from EL3.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

VBAR\_EL2 is a 64-bit register.

## Field descriptions

The VBAR\_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
Vector Base Address																															
Vector Base Address											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 [63:11]

Vector Base Address. Base address of the exception vectors for exceptions taken to EL2.

If the implementation does not support ARMv8.2-LVA, then:

- If tagged addresses are being used, bits [55:48] of VBAR\_EL2 must be 0 or else the use of the vector address will result in a recursive exception.
- If tagged addresses are not being used, bits [63:48] of VBAR\_EL2 must be 0 or else the use of the vector address will result in a recursive exception.

If the implementation supports ARMv8.2-LVA, then:

- If tagged addresses are being used, bits [55:52] of VBAR\_EL2 must be the same or else the use of the vector address will result in a recursive exception.
- If tagged addresses are not being used, bits [63:52] of VBAR\_EL2 must be the same or else the use of the vector address will result in a recursive exception.

### Bits [10:0]

Reserved, RES0.

## Accessing the VBAR\_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
VBAR_EL2	11	100	1100	0000	000
VBAR_EL1	11	000	1100	0000	000

## Accessibility

The register is accessible as follows:

<systemreg>	Control			Accessibility			
	E2H	TGE	NS	EL0	EL1	EL2	EL3
VBAR_EL2	x	x	0	-	-	n/a	RW
VBAR_EL2	0	0	1	-	-	RW	RW
VBAR_EL2	0	1	1	-	n/a	RW	RW
VBAR_EL2	1	0	1	-	-	RW	RW
VBAR_EL2	1	1	1	-	n/a	RW	RW
VBAR_EL1	x	x	0	-	<a href="#">VBAR_EL1</a>	n/a	<a href="#">VBAR_EL1</a>
VBAR_EL1	0	0	1	-	<a href="#">VBAR_EL1</a>	<a href="#">VBAR_EL1</a>	<a href="#">VBAR_EL1</a>
VBAR_EL1	0	1	1	-	n/a	<a href="#">VBAR_EL1</a>	<a href="#">VBAR_EL1</a>
VBAR_EL1	1	0	1	-	<a href="#">VBAR_EL1</a>	RW	<a href="#">VBAR_EL1</a>
VBAR_EL1	1	1	1	-	n/a	RW	<a href="#">VBAR_EL1</a>

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic VBAR\_EL2 or VBAR\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.



# VBAR\_EL3, Vector Base Address Register (EL3)

The VBAR\_EL3 characteristics are:

## Purpose

Holds the vector base address for any exception that is taken to EL3.

This register is part of:

- The Exception and fault handling registers functional group.
- The Security registers functional group.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

VBAR\_EL3 is a 64-bit register.

## Field descriptions

The VBAR\_EL3 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
Vector Base Address																															
Vector Base Address										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 [63:11]

Vector Base Address. Base address of the exception vectors for exceptions taken to EL3.

If the implementation does not support ARMv8.2-LVA, then:

- If tagged addresses are being used, bits [55:48] of VBAR\_EL3 must be 0 or else the use of the vector address will result in a recursive exception.
- If tagged addresses are not being used, bits [63:48] of VBAR\_EL3 must be 0 or else the use of the vector address will result in a recursive exception.

If the implementation supports ARMv8.2-LVA, then:

- If tagged addresses are being used, bits [55:52] of VBAR\_EL3 must be the same or else the use of the vector address will result in a recursive exception.
- If tagged addresses are not being used, bits [63:52] of VBAR\_EL3 must be the same or else the use of the vector address will result in a recursive exception.

### Bits [10:0]

Reserved, RES0.

## Accessing the VBAR\_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
VBAR_EL3	11	110	1100	0000	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
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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# VMPIDR\_EL2, Virtualization Multiprocessor ID Register

The VMPIDR\_EL2 characteristics are:

## Purpose

Holds the value of the Virtualization Multiprocessor ID. This is the value returned by Non-secure EL1 reads of [MPIDR\\_EL1](#).

This register is part of:

- The Identification registers functional group.
- The Virtualization registers functional group.

## Configuration

AArch64 System register VMPIDR\_EL2 bits [31:0] are architecturally mapped to AArch32 System register [VMPIDR](#).

If EL2 is not implemented, reads of this register return the value of the [MPIDR\\_EL1](#), and writes to the register are ignored.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

VMPIDR\_EL2 is a 64-bit register.

## Field descriptions

The VMPIDR\_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	0	0	0	0	Aff3							
1	U	0	0	0	0	0	MT	Aff2								Aff1								Aff0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:40]

Reserved, RES0.

### Aff3, bits [39:32]

Affinity level 3. Highest level affinity field.

### Bit [31]

Reserved, RES1.

### U, bit [30]

Indicates a Uniprocessor system, as distinct from PE 0 in a multiprocessor system. The possible values of this bit are:

U	Meaning
0	Processor is part of a multiprocessor system.
1	Processor is part of a uniprocessor system.

**Bits [29:25]**

Reserved, RES0.

**MT, bit [24]**

Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. The possible values of this bit are:

MT	Meaning
0	Performance of PEs at the lowest affinity level is largely independent.
1	Performance of PEs at the lowest affinity level is very interdependent.

**Aff2, bits [23:16]**

Affinity level 2. Second highest level affinity field.

**Aff1, bits [15:8]**

Affinity level 1. Third highest level affinity field.

**Aff0, bits [7:0]**

Affinity level 0. Lowest level affinity field.

**Accessing the VMPIDR\_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
VMPIDR_EL2	11	100	0000	0000	101

**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.

# 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 'Identification registers, functional group' in the ARMv8 ARM, section G4.18.1.

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.

# VTCR\_EL2, Virtualization Translation Control Register

The VTCR\_EL2 characteristics are:

## Purpose

The control register for stage 2 of the EL1&0 translation regime.

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

## Configuration

AArch64 System register VTCR\_EL2 is architecturally mapped to AArch32 System register [VTCR](#).

If EL2 is not implemented, this register is RES0 from EL3.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

VTCR\_EL2 is a 32-bit register.

## Field descriptions

The VTCR\_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
1	0	0	<a href="#">HWU62</a>	<a href="#">HWU61</a>	<a href="#">HWU60</a>	<a href="#">HWU59</a>	0	0	<a href="#">HDHA</a>	0	<a href="#">VS</a>		<a href="#">PS</a>		<a href="#">TG0</a>	<a href="#">SH0</a>	<a href="#">ORGN0</a>	<a href="#">IRGN0</a>	<a href="#">SL0</a>											<a href="#">T0SZ</a>	

Any of the bits in VTCR\_EL2 are permitted to be cached in a TLB.

### Bit [31]

Reserved, RES1.

### Bits [30:29]

Reserved, RES0.

### HWU62, bit [28]

In ARMv8.2:

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[62] of the stage 2 translation table block or level 3 entry.

Defined values are:

HWU62	Meaning
0	The stage 2 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 2 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.

This bit is RES0, if ARMv8.2-TTBBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU61, bit [27]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[61] of the stage 2 translation table block or level 3 entry.

Defined values are:

HWU61	Meaning
0	The stage 2 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 2 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.

This bit is RES0, if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU60, bit [26]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[60] of the stage 2 translation table block or level 3 entry.

Defined values are:

HWU60	Meaning
0	The stage 2 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 2 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**HWU59, bit [25]****In ARMv8.2:**

Hardware Use. Indicates IMPLEMENTATION DEFINED hardware use of bit[59] of the stage 2 translation table block or level 3 entry.

Defined values are:

HWU59	Meaning
0	The stage 2 translation table entry block or level 3 bit cannot be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.
1	The stage 2 translation table entry block or level 3 bit can be interpreted by hardware for an IMPLEMENTATION DEFINED purpose.

This bit is RES0 if ARMv8.2-TTPBHA is not implemented.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.



**Bits [24:23]**

Reserved, RES0.

**HD, bit [22]**

In ARMv8.2 and ARMv8.1:

Hardware management of dirty state in stage 2 translations from Non-secure EL0 and EL1.

Defined values are:

HD	Meaning
0	Stage 2 hardware management of dirty state disabled.
1	Stage 2 hardware management of dirty state enabled, only if the HA bit is also set to 1.

This bit is RES0 if ARMv8.1-TTBM is not implemented.

**In ARMv8.0:**

Reserved, RES0.

**HA, bit [21]**

In ARMv8.2 and ARMv8.1:

Hardware Access flag update in stage 2 translations from Non-secure EL0 and EL1.

Defined values are:

HA	Meaning
0	Stage 2 Access flag update disabled.
1	Stage 2 Access flag update enabled.

This bit is RES0 if ARMv8.1-TTBM is not implemented.

**In ARMv8.0:**

Reserved, RES0.

**Bit [20]**

Reserved, RES0.

**VS, bit [19]**

In ARMv8.2 and ARMv8.1:

VMID Size.

Defined values are:

VS	Meaning
0	8 bit - the upper 8 bits of <a href="#">VTTBR_EL2</a> are ignored by the hardware, and treated as if they are all zeros, for every purpose except when reading back the register.
1	16 bit - the upper 8 bits of <a href="#">VTTBR_EL2</a> are used for allocation and matching in the TLB.

If the implementation only supports an 8-bit VMID, this field is RES0.

This bit is RES0 if ARMv8.1-VMID16 is not implemented.

**In ARMv8.0:**

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 VTCR\_EL2 are 0000.

**TG0, bits [15:14]**

Granule size for the [VTTBR\\_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 [VTTBR\\_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 [VTTBR\\_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 [VTTBR\\_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

**SL0, bits [7:6]**

Starting level of the [VTCR\\_EL2](#) addressed region. The meaning of this field depends on the value of VTCR\_EL2.TG0 (the granule size).

SL0	Meaning
00	If TG0 is 00 (4KB granule), start at level 2. If TG0 is 10 (16KB granule) or 01 (64KB granule), start at level 3.
01	If TG0 is 00 (4KB granule), start at level 1. If TG0 is 10 (16KB granule) or 01 (64KB granule), start at level 2.
10	If TG0 is 00 (4KB granule), start at level 0. If TG0 is 10 (16KB granule) or 01 (64KB granule), start at level 1.

All other values are reserved. If this field is programmed to a reserved value, or to a value that is not consistent with the programming of T0SZ, then a stage 2 level 0 Translation fault is generated.

**T0SZ, bits [5:0]**

The size offset of the memory region addressed by [VTTBR\\_EL2](#). The region size is  $2^{(64-T0SZ)}$  bytes.

The maximum and minimum possible values for T0SZ depend on the level of translation table and the memory translation granule size, as described in the AArch64 Virtual Memory System Architecture chapter.

If this field is programmed to a value that is not consistent with the programming of SL0 then a stage 2 level 0 Translation fault is generated.

## Accessing the VTCR\_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
VTCR_EL2	11	100	0010	0001	010

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# VTTBR\_EL2, Virtualization Translation Table Base Register

The VTTBR\_EL2 characteristics are:

## Purpose

Holds the base address of the translation table for the initial lookup for stage 2 of an address translation in the Non-secure EL1&0 translation regime, and other information for this translation regime.

This register is part of:

- The Virtualization registers functional group.
- The Virtual memory control registers functional group.

## Configuration

AArch64 System register VTTBR\_EL2 is architecturally mapped to AArch32 System register [VTTBR](#).

If EL2 is not implemented, this register is RES0 from EL3.

RW fields in this register reset to architecturally UNKNOWN values.

## Attributes

VTTBR\_EL2 is a 64-bit register.

## Field descriptions

The VTTBR\_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
VMID[15:8]								VMID[7:0]								BADDR															
BADDR																															CnP
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Any of the fields in this register are permitted to be cached in a TLB.

### VMID[15:8], bits [63:56]

In ARMv8.2 and ARMv8.1:

Extension to VMID[7:0]. See VMID[7:0] for more details.

In ARMv8.0:

Reserved, RES0.

### VMID[7:0], bits [55:48]

The VMID for the translation table.

It is IMPLEMENTATION DEFINED whether the VMID is 8 bits or 16 bits.

If the implementation has an 8-bit VMID, then VMID[15:8] are RES0.

If the implementation has a 16-bit VMID, then:

- The [VTCR\\_EL2](#).VS bit selects whether VMID[15:8] are ignored by the hardware for every purpose except reading back the register, or whether these bits are used for allocation and matching in the TLB.

- The 16-bit VMID is only supported when EL2 is using AArch64. This means the hardware must ignore VMID[15:8] when EL2 is using AArch32.

**BADDR, bits [47:1]**

Translation table base address, A[47:x] or A[51:x], bits[47:1].

**Note**

- Translation table base addresses of 52 bits, A[51:x], are supported only in an implementation that includes ARMv8.2-LPA and is using the 64KB translation granule.
- A translation table must be aligned to the size of the table, except that when using a translation table base address larger than 48 bits the minimum alignment of a table containing fewer than eight entries is 64 bytes.

In an implementation that includes ARMv8.2-LPA, if the value of [VTCTR\\_EL2](#).IPS is 110, then:

- Register bits[47:z] hold bits[47:z] of the stage 1 translation table base address, where z is determined as follows:
  - If  $x \geq 6$  then  $z=x$ .
  - Otherwise,  $z=6$ .
- Register bits[5:2] hold bits[51:48] of the stage 1 translation table base address.
- When  $z > x$  register bits[(z-1):x] are RES0, and bits[(z-1):x] of the translation table base address are zero.
- When  $x > 6$  register bits[(x-1):6] are RES0.
- Register bit[1] is RES0.
- Bits[5:2] of the stage 1 translation table base address are zero.
- In an implementation that includes ARMv8.2-TTCNP bit[0] of the stage 1 translation table base address is zero.

**Note**

In an implementation that includes ARMv8.2-LPA a [VTCTR\\_EL2](#).PS value of 110, that selects a PA size of 52 bits, is permitted only when using the 64KB translation granule.

When the value of [ID\\_AA64MMFR0\\_EL1](#).PARange indicates that the implementation does not support a 52 bit PA size, if a translation table lookup uses this register with the 64KB translation granule when the value of [VTCTR\\_EL2](#).PS is 110 and the value of register bits[5:2] is nonzero it is IMPLEMENTATION DEFINED whether an Address size fault is generated, but ARM deprecates not generating an Address size fault.

If the Effective value of [VTCTR\\_EL2](#).PS is not 110 then:

- Register bits[47:x] hold bits[47:x] of the stage 1 translation table base address.
- Register bits[(x-1):1] are RES0.
- If the implementation supports 52-bit PAs and IPAs then bits[51:49] of the translation table base addresses used in this stage of translation are 0b0000.

**Note**

This definition applies:

- To an implementation that includes ARMv8.2-LPA and is using a translation granule smaller than 64KB.
- To any implementation that does not include ARMv8.2-LPA.

If any VTTBR\_EL2[47:0] bit that is defined as RES0 has the value 1 when a translation table walk is performed using VTTBR\_EL2, then the translation table base address might be misaligned, with effects that are CONSTRAINED UNPREDICTABLE, and must be one of the following:

- Bits[x-1:0] of the translation table base address are treated as if all the bits are zero. The value read back from the corresponding register bits is either the value written to the register or zero.
- The result of the calculation of an address for a translation table walk using this register can be corrupted in those bits that are nonzero.

The AArch64 Virtual Memory System Architecture chapter describes how x is calculated based on the value of [VTCTR\\_EL2](#).T0SZ, the stage of translation, and the translation granule size.

**CnP, bit [0]****In ARMv8.2:**

Common not Private. In an implementation that includes ARMv8.2-TTCNP, indicates whether each entry that is pointed to by VTTBR\_EL2 is a member of a common set that can be used by every PE in the Inner Shareable domain for which the value of VTTBR\_EL2.CnP is 1.

CnP	Meaning
0	The translation table entries pointed to by VTTBR_EL2 are permitted to differ from the entries for VTTBR_EL2 for other PEs in the Inner Shareable domain. This is not affected by the value of the current VMID.
1	The translation table entries pointed to by VTTBR_EL2 are the same as the translation table entries for every other PE in the Inner Shareable domain for which the value of VTTBR_EL2.CnP is 1 and the VMID is the same as the current VMID.

**Note**

If the value of VTTBR\_EL2.CnP bit is 1 on multiple PEs in the same Inner Shareable domain and those VTTBR\_EL2s do not point to the same translation table entries when using the current VMID then the results of translations using VTTBR\_EL2 are CONstrained UNPREDICTABLE, see 'CONstrained UNPREDICTABLE behaviors due to caching of control or data values' in the ARMv8-A ARM appendix K1.

In an implementation that does not include ARMv8.2-TTCNP this field is RES0.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**Accessing the VTTBR\_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
VTTBR_EL2	11	100	0010	0001	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.

# 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 C1SW](#): Data or unified Cache line Clean by Set/Way

[DC C1VAC](#): Data or unified Cache line Clean by VA to PoC

[DC C1VAP](#): Data or unified Cache line Clean by VA to PoP

[DC C1VAU](#): 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, EL1, Last Level, 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

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S12E0R, Address Translate Stages 1 and 2 EL0 Read

The AT S12E0R characteristics are:

## Purpose

Performs stage 1 and 2 address translations, with permissions as if reading from the given virtual address, using the following translation regime:

- In Secure state, and in Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime.
- In Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

AT S12E0R is a 64-bit System instruction.

## Field descriptions

The AT S12E0R 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
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S12E0R instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S12E0R	01	100	0111	1000	110

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	WO	WO

0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, or stage 2 translation is disabled, or the instruction is executed at EL3 when the value of [SCR\\_EL3.NS](#) is 0, this instruction executes as [AT S1E0R](#).

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S12E0W, Address Translate Stages 1 and 2 EL0 Write

The AT S12E0W characteristics are:

## Purpose

Performs stage 1 and 2 address translations, with permissions as if writing to the given virtual address, using the following translation regime:

- In Secure state, and in Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime.
- In Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

AT S12E0W is a 64-bit System instruction.

## Field descriptions

The AT S12E0W 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
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Input address for translation																															

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S12E0W instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S12E0W	01	100	0111	1000	111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	WO	WO

0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, or stage 2 translation is disabled, or the instruction is executed at EL3 when the value of [SCR\\_EL3.NS](#) is 0, this instruction executes as [AT S1E0W](#).

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S12E1R, Address Translate Stages 1 and 2 EL1 Read

The AT S12E1R characteristics are:

## Purpose

Performs stage 1 and 2 address translation, with permissions as if reading from the given virtual address, using the following translation regime:

- In Secure state, and in Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime.
- In Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

AT S12E1R is a 64-bit System instruction.

## Field descriptions

The AT S12E1R 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
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S12E1R instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S12E1R	01	100	0111	1000	100

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	WO	WO

0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, or stage 2 translation is disabled, or the instruction is executed at EL3 when the value of [SCR\\_EL3.NS](#) is 0, this instruction executes as [AT S1E1R](#).

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S12E1W, Address Translate Stages 1 and 2 EL1 Write

The AT S12E1W characteristics are:

## Purpose

Performs stage 1 and 2 address translation, with permissions as if writing to the given virtual address, using the following translation regime:

- In Secure state, and in Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime.
- In Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

AT S12E1W is a 64-bit System instruction.

## Field descriptions

The AT S12E1W 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
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S12E1W instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S12E1W	01	100	0111	1000	101

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	WO	WO



0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, or stage 2 translation is disabled, or the instruction is executed at EL3 when the value of [SCR\\_EL3.NS](#) is 0, this instruction executes as [AT S1E1W](#).

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S1E0R, Address Translate Stage 1 EL0 Read

The AT S1E0R characteristics are:

## Purpose

Performs stage 1 address translation, with permissions as if reading from the given virtual address, using the following translation regime:

- In Secure state, and in Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime.
- In Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

AT S1E0R is a 64-bit System instruction.

## Field descriptions

The AT S1E0R 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
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S1E0R instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S1E0R	01	000	0111	1000	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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S1E0W, Address Translate Stage 1 EL0 Write

The AT S1E0W characteristics are:

## Purpose

Performs stage 1 address translation, with permissions as if writing to the given virtual address, using the following translation regime:

- In Secure state, and in Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime.
- In Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

AT S1E0W is a 64-bit System instruction.

## Field descriptions

The AT S1E0W 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
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Input address for translation																															

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S1E0W instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S1E0W	01	000	0111	1000	011

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S1E1R, Address Translate Stage 1 EL1 Read

The AT S1E1R characteristics are:

## Purpose

Performs stage 1 address translation, with permissions as if reading from the given virtual address, using the following translation regime:

- In Secure state, and in Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime.
- In Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

AT S1E1R is a 64-bit System instruction.

## Field descriptions

The AT S1E1R 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
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S1E1R instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S1E1R	01	000	0111	1000	000

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S1E1RP, Address Translate Stage 1 EL1 Read PAN

The AT S1E1RP characteristics are:

## Purpose

When ARMv8.2-ATS1E1 is implemented, performs a stage 1 address translation, where the value of PSTATE.PAN determines if a read from a location will generate a permission fault for a privileged access, using the following translation regime:

- In Secure state, and in Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime.
- In Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime.

This System instruction is part of the Address translation instructions functional group.

## Configuration

This instruction is introduced in ARMv8.2.

## Attributes

AT S1E1RP is a 64-bit System instruction.

## Field descriptions

The AT S1E1RP 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
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S1E1RP instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S1E1RP	01	000	0111	1001	000

## 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S1E1W, Address Translate Stage 1 EL1 Write

The AT S1E1W characteristics are:

## Purpose

Performs stage 1 address translation, with permissions as if writing to the given virtual address, using the following translation regime:

- In Secure state, and in Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime.
- In Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

AT S1E1W is a 64-bit System instruction.

## Field descriptions

The AT S1E1W 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
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S1E1W instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S1E1W	01	000	0111	1000	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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S1E1WP, Address Translate Stage 1 EL1 Write PAN

The AT S1E1WP characteristics are:

## Purpose

When ARMv8.2-ATS1E1 is implemented, performs a stage 1 address translation, where the value of PSTATE.PAN determines if a write to a location will generate a permission fault for a privileged access, using the following translation regime:

- In Secure state, and in Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is not {1, 1}, the EL1&0 translation regime.
- In Non-secure state when [HCR\\_EL2](#).{E2H, TGE} is {1, 1}, the EL2&0 translation regime.

This System instruction is part of the Address translation instructions functional group.

## Configuration

This instruction is introduced in ARMv8.2.

## Attributes

AT S1E1WP is a 64-bit System instruction.

## Field descriptions

The AT S1E1WP 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
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S1E1WP instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S1E1WP	01	000	0111	1001	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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S1E2R, Address Translate Stage 1 EL2 Read

The AT S1E2R characteristics are:

## Purpose

Performs stage 1 address translation as defined for EL2, with permissions as if reading from the given virtual address.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

AT S1E2R is a 64-bit System instruction.

## Field descriptions

The AT S1E2R 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
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S1E2R instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S1E2R	01	100	0111	1000	000

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	-
0	0	1	-	-	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO

1	1	1	-	n/a	WO	WO
---	---	---	---	-----	----	----

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED at EL3.

# AT S1E2W, Address Translate Stage 1 EL2 Write

The AT S1E2W characteristics are:

## Purpose

Performs stage 1 address translation as defined for EL2, with permissions as if writing to the given virtual address.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

AT S1E2W is a 64-bit System instruction.

## Field descriptions

The AT S1E2W 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
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S1E2W instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S1E2W	01	100	0111	1000	001

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	-
0	0	1	-	-	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO



1	1	1	-	n/a	WO	WO
---	---	---	---	-----	----	----

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED at EL3.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S1E3R, Address Translate Stage 1 EL3 Read

The AT S1E3R characteristics are:

## Purpose

Performs stage 1 address translation as defined for EL3, with permissions as if reading from the given virtual address.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

AT S1E3R is a 64-bit System instruction.

## Field descriptions

The AT S1E3R 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
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S1E3R instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S1E3R	01	110	0111	1000	000

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	-	WO
0	1	1	-	n/a	-	WO
1	0	1	-	-	-	WO

1	1	1	-	n/a	-	WO
---	---	---	---	-----	---	----

This table applies to all syntax that can be used to execute this instruction.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# AT S1E3W, Address Translate Stage 1 EL3 Write

The AT S1E3W characteristics are:

## Purpose

Performs stage 1 address translation as defined for EL3, with permissions as if writing to the given virtual address.

This System instruction is part of the Address translation instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

AT S1E3W is a 64-bit System instruction.

## Field descriptions

The AT S1E3W 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
Input address for translation																															
Input address for translation																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Input address for translation. The resulting address can be read from the [PAR\\_EL1](#).

If the address translation instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then VA[63:32] is RES0.

## Executing the AT S1E3W instruction

This instruction is executed using AT with the following syntax:

AT <at\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<at_op>	op0	op1	CRn	CRm	op2
S1E3W	01	110	0111	1000	001

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	-	WO
0	1	1	-	n/a	-	WO
1	0	1	-	-	-	WO

1	1	1	-	n/a	-	WO
---	---	---	---	-----	---	----

This table applies to all syntax that can be used to execute this instruction.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

## Purpose

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

## Attributes

## Field descriptions

63	62	61	60	59	58	57	56	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
SetWay																												Level		0	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**Bits [63:32]**

### SetWay, bits [31:4]

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

**Level, bits [3:1]**

**Bit [0]**

Page 1626

## Executing the DC CISC instruction

This instruction is executed using DC with the following syntax:

DC <dc\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<dc_op>	op0	op1	CRn	CRm	op2
CISC	01	000	0111	1110	010

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is CONSTRAINED UNPREDICTABLE and one of the following occurs:

- The instruction is UNDEFINED
- The instruction performs cache maintenance on one of:
  - No cache lines.
  - A single arbitrary cache line.
  - Multiple arbitrary cache lines.

## 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.TSW==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.TSW==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

# DC CIVAC, Data or unified Cache line Clean and Invalidate by VA to PoC

The DC CIVAC characteristics are:

## Purpose

Clean and Invalidate data cache by address to Point of Coherency.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch64 System instruction DC CIVAC performs the same function as AArch32 System instruction [DCCIMVAC](#).

## Attributes

DC CIVAC is a 64-bit System instruction.

## Field descriptions

The DC CIVAC 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
Virtual address to use																															
Virtual address to use																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Virtual address to use.

## Executing the DC CIVAC instruction

This instruction is executed using DC with the following syntax:

DC <dc\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<dc_op>	op0	op1	CRn	CRm	op2
CIVAC	01	011	0111	1110	001

## Accessibility

The instruction is executable 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 syntax that can be used to execute this instruction.



If EL0 access is enabled, when executing at EL0, this instruction requires read access permission to the VA, otherwise it causes a Permission Fault.

## 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.

In both Security states, and not dependent on other configuration bits:

- If [SCTLR\\_EL1](#).UCI==0, execution of this instruction at EL0 is trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HCR\\_EL2](#).TPC==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HCR\\_EL2](#).TPC==1, and [SCTLR\\_EL1](#).UCI==1, Non-secure execution of this instruction at EL0 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](#).TPC==1, Non-secure execution of this instruction at EL0 is trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DC CSW, Data or unified Cache line Clean by Set/Way

The DC CSW characteristics are:

## Purpose

Clean data cache by set/way.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch64 System instruction DC CSW performs the same function as AArch32 System instruction [DCCSW](#).

## Attributes

DC CSW is a 64-bit System instruction.

## Field descriptions

The DC CSW 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
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
SetWay																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:32]

Reserved, RES0.

### SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$ ,  $L = \text{Log}_2(\text{LINELEN})$ ,  $B = (L + S)$ ,  $S = \text{Log}_2(\text{NSETS})$ .

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

### Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

### Bit [0]

Reserved, RES0.

## Executing the DC CSW instruction

This instruction is executed using DC with the following syntax:

DC <dc\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<dc_op>	op0	op1	CRn	CRm	op2
CSW	01	000	0111	1010	010

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is **CONSTRAINED UNPREDICTABLE** and one of the following occurs:

- The instruction is **UNDEFINED**
- The instruction performs cache maintenance on one of:
  - No cache lines.
  - A single arbitrary cache line.
  - Multiple arbitrary cache lines.

## 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](#).TSW==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](#).TSW==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DC CVAC, Data or unified Cache line Clean by VA to PoC

The DC CVAC characteristics are:

## Purpose

Clean data cache by address to Point of Coherency.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch64 System instruction DC CVAC performs the same function as AArch32 System instruction [DCCMVAC](#).

## Attributes

DC CVAC is a 64-bit System instruction.

## Field descriptions

The DC CVAC 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
Virtual address to use																															
Virtual address to use																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Virtual address to use.

## Executing the DC CVAC instruction

This instruction is executed using DC with the following syntax:

DC <dc\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<dc_op>	op0	op1	CRn	CRm	op2
CVAC	01	011	0111	1010	001

## Accessibility

The instruction is executable 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 syntax that can be used to execute this instruction.

If EL0 access is enabled, when executing at EL0, this instruction requires read access permission to the VA, otherwise it causes a Permission Fault.

## 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.

In both Security states, and not dependent on other configuration bits:

- If [SCTLR\\_EL1](#).UCI==0, execution of this instruction at EL0 is trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HCR\\_EL2](#).TPC==1, Non-secure execution of this instruction at EL1 is trapped to EL2.
- If [HCR\\_EL2](#).TPC==1, and [SCTLR\\_EL1](#).UCI==1, Non-secure execution of this instruction at EL0 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](#).TPC==1, Non-secure execution of this instruction at EL0 is trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DC CVAP, Data or unified Cache line Clean by VA to PoP

The DC CVAP characteristics are:

## Purpose

Clean data cache by address to Point of Persistence.

If the memory system does not identify a Point of Persistence, then this instruction behaves as a [DC CVAC](#).

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

This instruction is introduced in ARMv8.2.

## Attributes

DC CVAP is a 64-bit System instruction.

## Field descriptions

The DC CVAP 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
Virtual address to use																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Virtual address to use.

## Executing the DC CVAP instruction

This instruction is executed using DC with the following syntax:

DC <dc\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<dc_op>	op0	op1	CRn	CRm	op2
CVAP	01	011	0111	1100	001

## Accessibility

The instruction is executable 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 syntax that can be used to execute this instruction.

If EL0 access is enabled, when executing at EL0, this instruction requires read access permission to the VA, otherwise it causes a Permission Fault.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DC CVAU, Data or unified Cache line Clean by VA to PoU

The DC CVAU characteristics are:

## Purpose

Clean data cache by address to Point of Unification.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch64 System instruction DC CVAU performs the same function as AArch32 System instruction [DCCMVAU](#).

## Attributes

DC CVAU is a 64-bit System instruction.

## Field descriptions

The DC CVAU 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
Virtual address to use																															
Virtual address to use																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Virtual address to use.

## Executing the DC CVAU instruction

This instruction is executed using DC with the following syntax:

DC <dc\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<dc_op>	op0	op1	CRn	CRm	op2
CVAU	01	011	0111	1011	001

## Accessibility

The instruction is executable 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 syntax that can be used to execute this instruction.



If EL0 access is enabled, when executing at EL0, this instruction requires read access permission to the VA, otherwise it causes a Permission Fault.

## 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.

In both Security states, and not dependent on other configuration bits:

- If [SCTLR\\_EL1](#).UCI==0, execution of this instruction at EL0 is trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HCR\\_EL2](#).TPU==1, Non-secure execution of this instruction at EL0 and 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](#).TPU==1, Non-secure execution of this instruction at EL0 and EL1 is trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DC ISW, Data or unified Cache line Invalidate by Set/Way

The DC ISW characteristics are:

## Purpose

Invalidate data cache by set/way.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch64 System instruction DC ISW performs the same function as AArch32 System instruction [DCISW](#).

## Attributes

DC ISW is a 64-bit System instruction.

## Field descriptions

The DC ISW 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
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
SetWay																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:32]

Reserved, RES0.

### SetWay, bits [31:4]

Contains two fields:

- Way, bits[31:32-A], the number of the way to operate on.
- Set, bits[B-1:L], the number of the set to operate on.

Bits[L-1:4] are RES0.

$A = \text{Log}_2(\text{ASSOCIATIVITY})$ ,  $L = \text{Log}_2(\text{LINELEN})$ ,  $B = (L + S)$ ,  $S = \text{Log}_2(\text{NSETS})$ .

ASSOCIATIVITY, LINELEN (line length, in bytes), and NSETS (number of sets) have their usual meanings and are the values for the cache level being operated on. The values of A and S are rounded up to the next integer.

### Level, bits [3:1]

Cache level to operate on, minus 1. For example, this field is 0 for operations on L1 cache, or 1 for operations on L2 cache.

### Bit [0]

Reserved, RES0.

## Executing the DC ISW instruction

This instruction is executed using DC with the following syntax:

DC <dc\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<dc_op>	op0	op1	CRn	CRm	op2
ISW	01	000	0111	0110	010

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

At EL1, this instruction performs a cache clean and invalidate, meaning it performs the same invalidation as a [DC C1SW](#) instruction, if all of the following apply:

- EL2 is implemented.
- The value of [HCR\\_EL2.SWIO](#) is 1 or the value of [HCR\\_EL2.VM](#) is 1.
- The value of [SCR\\_EL3.NS](#) is 1 or EL3 is not implemented.

If this instruction is executed with a set, way or level argument that is larger than the value supported by the implementation then the behavior is CONSTRAINED UNPREDICTABLE and one of the following occurs:

- The instruction is UNDEFINED
- The instruction performs cache maintenance on one of:
  - No cache lines.
  - A single arbitrary cache line.
  - Multiple arbitrary cache lines.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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.TSW](#)==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.TSW](#)==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

# DC IVAC, Data or unified Cache line Invalidate by VA to PoC

The DC IVAC characteristics are:

## Purpose

Invalidate data cache by address to Point of Coherency.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch64 System instruction DC IVAC performs the same function as AArch32 System instruction [DCIMVAC](#).

## Attributes

DC IVAC is a 64-bit System instruction.

## Field descriptions

The DC IVAC 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
Virtual address to use																															
Virtual address to use																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Virtual address to use.

## Executing the DC IVAC instruction

This instruction is executed using DC with the following syntax:

DC <dc\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<dc_op>	op0	op1	CRn	CRm	op2
IVAC	01	000	0111	0110	001

When the instruction is executed, it can generate a watchpoint, which is prioritized in the same way as other watchpoints. If a watchpoint is generated, the CM bit in the [ESR\\_ELx](#).ISS field is set to 1.

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

This instruction requires write access permission to the VA, otherwise it causes a Permission Fault.

At EL1, this instruction performs a cache clean and invalidate, meaning it performs the same invalidation as a [DC CIVAC](#) instruction, if all of the following apply:

- EL2 is implemented.
- [HCR\\_EL2](#).VM is set to 1.
- [SCR\\_EL3](#).NS is set to 1 or EL3 is not implemented.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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](#).TPC=1, Non-secure execution of this instruction at EL1 is trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DC ZVA, Data Cache Zero by VA

The DC ZVA characteristics are:

## Purpose

Zero data cache by address. Zeroes a naturally aligned block of N bytes, where the size of N is identified in [DCZID\\_EL0](#).

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

DC ZVA is a 64-bit System instruction.

## Field descriptions

The DC ZVA 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
Virtual address to use																															
Virtual address to use																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Virtual address to use. There is no alignment restriction on the address within the block of N bytes that is used.

## Executing the DC ZVA instruction

This instruction is executed using DC with the following syntax:

DC <dc\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<dc_op>	op0	op1	CRn	CRm	op2
ZVA	01	011	0111	0100	001

When this instruction is executed, it can generate memory faults or watchpoints which are prioritized in the same way as other memory-related faults or watchpoints. If a synchronous data abort fault or a watchpoint is generated, the CM bit in the [ESR\\_ELx.ISS](#) field is not set.

If the memory region being zeroed is any type of Device memory, this instruction can give an alignment fault which is prioritized in the same way as other alignment faults that are determined by the memory type.

This instruction applies to Normal memory regardless of cacheability attributes.

This instruction behaves as a set of Stores to each byte within the block being accessed, and so it:

- Generates a Permission Fault if the translation system does not permit writes to the locations.
- Requires the same considerations for ordering and the management of coherency as any other store instructions.

## Accessibility

The instruction is executable 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 syntax that can be used to execute this instruction.

When EL0 access is disabled ([SCTLR\\_EL1](#).DZE is set to 0) and [HCR\\_EL2](#).TDZ is set to 1, execution of this instruction at EL0 is UNDEFINED.

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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.

In both Security states, and not dependent on other configuration bits:

- If [SCTLR\\_EL1](#).DZE==0, execution of this instruction at EL0 is trapped to EL1.

When EL2 is implemented and is using AArch64 and [SCR\\_EL3](#).NS==1 && [HCR\\_EL2](#).E2H==0 :

- If [HCR\\_EL2](#).TDZ==1, Non-secure execution of this instruction at EL0 and 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](#).TDZ==1, Non-secure execution of this instruction at EL0 and EL1 is trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# IC IALLU, Instruction Cache Invalidate All to PoU

The IC IALLU characteristics are:

## Purpose

Invalidate all instruction caches to Point of Unification.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch64 System instruction IC IALLU performs the same function as AArch32 System instruction [ICIALLU](#).

## Attributes

IC IALLU is a 64-bit System instruction.

## Field descriptions

IC IALLU ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the IC IALLU instruction

This instruction is executed using IC with the following syntax:

IC <ic\_op>

This syntax uses the following encoding in the System instruction encoding space:

<ic_op>	op0	op1	CRn	CRm	op2	Rt
IALLU	01	000	0111	0101	000	1111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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 [IC IALLUIS](#).

## 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](#).TPU==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](#).TPU==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# IC IALLUIS, Instruction Cache Invalidate All to PoU, Inner Shareable

The IC IALLUIS characteristics are:

## Purpose

Invalidate all instruction caches in Inner Shareable domain to Point of Unification.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch64 System instruction IC IALLUIS performs the same function as AArch32 System instruction [ICIALLUIS](#).

## Attributes

IC IALLUIS is a 64-bit System instruction.

## Field descriptions

IC IALLUIS ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the IC IALLUIS instruction

This instruction is executed using IC with the following syntax:

IC <ic\_op>

This syntax uses the following encoding in the System instruction encoding space:

<ic_op>	op0	op1	CRn	CRm	op2	Rt
IALLUIS	01	000	0111	0001	000	11111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	WO	n/a	WO
x	0	1	-	WO	WO	WO
x	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](#).TPU==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](#).TPU==1, Non-secure execution of this instruction at EL1 is trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# IC IVAU, Instruction Cache line Invalidate by VA to PoU

The IC IVAU characteristics are:

## Purpose

Invalidate instruction cache by address to Point of Unification.

This System instruction is part of the Cache maintenance instructions functional group.

## Configuration

AArch64 System instruction IC IVAU performs the same function as AArch32 System instruction [ICIMVAU](#).

## Attributes

IC IVAU is a 64-bit System instruction.

## Field descriptions

The IC IVAU 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
Virtual address to use																															
Virtual address to use																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Virtual address to use.

## Executing the IC IVAU instruction

This instruction is executed using IC with the following syntax:

IC <ic\_op>, <Xt>

This syntax uses the following encoding in the System instruction encoding space:

<ic_op>	op0	op1	CRn	CRm	op2
IVAU	01	011	0111	0101	001

## Accessibility

The instruction is executable 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 syntax that can be used to execute this instruction.

If EL0 access is enabled, when executing at EL0, this instruction requires read access permission to the VA, otherwise it is IMPLEMENTATION DEFINED whether it causes a Permission Fault.

## 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.

In both Security states, and not dependent on other configuration bits:

- If [SCTLR\\_EL1](#).UCI==0, execution of this instruction at EL0 is trapped to EL1.

When EL2 is implemented and is using AArch64 and SCR\_EL3.NS==1 && HCR\_EL2.E2H==0 :

- If [HCR\\_EL2](#).TPU==1, Non-secure execution of this instruction at EL0 and 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](#).TPU==1, Non-secure execution of this instruction at EL0 and EL1 is trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# S1\_<op1>\_<Cn>\_<Cm>\_<op2>, IMPLEMENTATION DEFINED maintenance instructions

The S1\_<op1>\_<Cn>\_<Cm>\_<op2> characteristics are:

## Purpose

This area of the System instruction encoding space is reserved for IMPLEMENTATION DEFINED System instructions.

This System instruction is part of the IMPLEMENTATION DEFINED functional group.

## Configuration

There are no configuration notes.

## Attributes

S1\_<op1>\_<Cn>\_<Cm>\_<op2> is a 64-bit System instruction.

## Field descriptions

The S1\_<op1>\_<Cn>\_<Cm>\_<op2> 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
IMPLEMENTATION DEFINED																															
IMPLEMENTATION DEFINED																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### IMPLEMENTATION DEFINED, bits [63:0]

IMPLEMENTATION DEFINED.

## Executing the S1\_<op1>\_<Cn>\_<Cm>\_<op2> instruction

This instruction is executed using SYS with the following syntax:

```
SYS <op1>, C<Cn>, C<Cm>, <op2>
```

This instruction is executed using SYSL with the following syntax:

```
SYSL <op1>, C<Cn>, C<Cm>, <op2>
```

This syntax uses the following encoding in the System instruction encoding space:

CRn	op1	op2	CRm
Cn<3:0>	op1<2:0>	op2<2:0>	Cm<3:0>

The value of <Cn> must be either 11 or 15. Other values may refer to architecturally-defined system instructions.

## Accessibility

The accessibility of system instructions with these encodings is IMPLEMENTATION DEFINED.

## 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.TIDCP==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.TIDCP==1](#), Non-secure execution of this instruction at EL1 is trapped to EL2.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI ALLE1, TLB Invalidate All, EL1

The TLBI ALLE1 characteristics are:

## Purpose

Invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 or stage 2 translation table entry, from any level of the translation table walk.
- 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, the entry would be required to translate an address using the Non-secure EL1&0 translation regime.

The invalidation applies to entries with any VMID.

The invalidation only applies to the PE that executes this instruction.

---

### Note

For the EL1&0 translation regime, the invalidation applies to both:

- Global entries.
  - Non-global entries with any ASID.
- 

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI ALLE1 is a 64-bit System instruction.

## Field descriptions

TLBI ALLE1 ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBI ALLE1 instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2	Rt
ALLE1	01	100	1000	0111	100	1111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	WO	WO



0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI ALLE1IS, TLB Invalidate All, EL1, Inner Shareable

The TLBI ALLE1IS characteristics are:

## Purpose

Invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 or stage 2 translation table entry, from any level of the translation table walk.
- 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, the entry would be required to translate an address using the Non-secure EL1&0 translation regime.

The invalidation applies to entries with any VMID.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this instructions.

---

### Note

For the EL1&0 translation regime, the invalidation applies to both:

- Global entries.
  - Non-global entries with any ASID.
- 

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI ALLE1IS is a 64-bit System instruction.

## Field descriptions

TLBI ALLE1IS ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBI ALLE1IS instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2	Rt
ALLE1IS	01	100	1000	0011	100	11111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	WO	WO

0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI ALLE2, TLB Invalidate All, EL2

The TLBI ALLE2 characteristics are:

## Purpose

If EL2 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- SCR\_EL3.NS is 1.
- The entry would be required to translate an address using the EL2 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 ALLE2 is a 64-bit System instruction.

## Field descriptions

TLBI ALLE2 ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBI ALLE2 instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2	Rt
ALLE2	01	100	1000	0111	000	1111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	-
0	0	1	-	-	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.



# TLBI ALLE2IS, TLB Invalidate All, EL2, Inner Shareable

The TLBI ALLE2IS characteristics are:

## Purpose

If EL2 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- SCR\_EL3.NS is 1.
- The entry would be required to translate an address using the EL2 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 ALLE2IS is a 64-bit System instruction.

## Field descriptions

TLBI ALLE2IS ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBI ALLE2IS instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2	Rt
ALLE2IS	01	100	1000	0011	000	1111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	-
0	0	1	-	-	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, this instruction is UNDEFINED.



# TLBI ALLE3, TLB Invalidate All, EL3

The TLBI ALLE3 characteristics are:

## Purpose

If EL3 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- The entry would be required to translate an address using the EL3 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 ALLE3 is a 64-bit System instruction.

## Field descriptions

TLBI ALLE3 ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBI ALLE3 instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2	Rt
ALLE3	01	110	1000	0111	000	1111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	-	WO
0	1	1	-	n/a	-	WO
1	0	1	-	-	-	WO
1	1	1	-	n/a	-	WO

This table applies to all syntax that can be used to execute this instruction.





# TLBI ALLE3IS, TLB Invalidate All, EL3, Inner Shareable

The TLBI ALLE3IS characteristics are:

## Purpose

If EL3 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- The entry would be required to translate an address using the EL3 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 ALLE3IS is a 64-bit System instruction.

## Field descriptions

TLBI ALLE3IS ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBI ALLE3IS instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2	Rt
ALLE3IS	01	110	1000	0011	000	11111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	-	WO
0	1	1	-	n/a	-	WO
1	0	1	-	-	-	WO
1	1	1	-	n/a	-	WO

This table applies to all syntax that can be used to execute this instruction.



# 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<sup>®</sup> 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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<sup>®</sup> 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.

# TLBI IPAS2E1, TLB Invalidate by Intermediate Physical Address, Stage 2, EL1

The TLBI IPAS2E1 characteristics are:

## Purpose

If EL2 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from any level of the translation table walk.
- SCR\_EL3.NS is 1.
- The entry would be used with the current VMID.
- The entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation only applies to the PE that executes this instruction.

For more information about the architectural requirements for this instruction see 'Invalidation of TLB entries from stage 2 translations' in the ARMv8 ARM.

This System instruction is part of:

- The TLB maintenance instructions functional group.
- The Virtualization registers functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI IPAS2E1 is a 64-bit System instruction.

## Field descriptions

The TLBI IPAS2E1 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IPA[51:48]				IPA[47:12]						
IPA[47:12]																																		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			

### Bits [63:40]

Reserved, RES0.

### IPA[51:48], bits [39:36]

In ARMv8.2:

Extension to IPA[47:12]. See IPA[47:12] for more details.

### In ARMv8.1 and ARMv8.0:

Reserved, RES0.



**IPA[47:12], bits [35:0]**

Bits[47:12] of the intermediate physical address to match.

When ARMv8.2-LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

**Executing the TLBI IPAS2E1 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
IPAS2E1	01	100	1000	0100	001

**Accessibility**

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
x	0	1	-	-	WO	WO
x	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, or SCR\_EL3.NS is 0, this instruction is a NOP.

# TLBI IPAS2E1IS, TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable

The TLBI IPAS2E1IS characteristics are:

## Purpose

If EL2 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from any level of the translation table walk.
- SCR\_EL3.NS is 1.
- The entry would be used with the current VMID.
- The entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this instructions.

For more information about the architectural requirements for this instruction see 'Invalidation of TLB entries from stage 2 translations' in the ARMv8 ARM.

This System instruction is part of:

- The TLB maintenance instructions functional group.
- The Virtualization registers functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI IPAS2E1IS is a 64-bit System instruction.

## Field descriptions

The TLBI IPAS2E1IS 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
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
IPA[47:12]																								IPA[51:48]				IPA[47:12]			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:40]

Reserved, RES0.

### IPA[51:48], bits [39:36]

In ARMv8.2:

Extension to IPA[47:12]. See IPA[47:12] for more details.

### In ARMv8.1 and ARMv8.0:

Reserved, RES0.

**IPA[47:12], bits [35:0]**

Bits[47:12] of the intermediate physical address to match.

When ARMv8.2-LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

**Executing the TLBI IPAS2E1IS 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
IPAS2E1IS	01	100	1000	0000	001

**Accessibility**

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
x	0	1	-	-	WO	WO
x	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, or SCR\_EL3.NS is 0, this instruction is a NOP.

# TLBI IPAS2LE1, TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1

The TLBI IPAS2LE1 characteristics are:

## Purpose

If EL2 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from the final level of the translation table walk.
- SCR\_EL3.NS is 1.
- The entry would be used with the current VMID.
- The entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation only applies to the PE that executes this instruction.

For more information about the architectural requirements for this instruction see 'Invalidation of TLB entries from stage 2 translations' in the ARMv8 ARM.

This System instruction is part of:

- The TLB maintenance instructions functional group.
- The Virtualization registers functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI IPAS2LE1 is a 64-bit System instruction.

## Field descriptions

The TLBI IPAS2LE1 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IPA[51:48]				IPA[47:12]			
IPA[47:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:40]

Reserved, RES0.

### IPA[51:48], bits [39:36]

In ARMv8.2:

Extension to IPA[47:12]. See IPA[47:12] for more details.

### In ARMv8.1 and ARMv8.0:

Reserved, RES0.

**IPA[47:12], bits [35:0]**

Bits[47:12] of the intermediate physical address to match.

When ARMv8.2-LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

**Executing the TLBI IPAS2LE1 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
IPAS2LE1	01	100	1000	0100	101

**Accessibility**

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
x	0	1	-	-	WO	WO
x	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, or SCR\_EL3.NS is 0, this instruction is a NOP.

# TLBI IPAS2LE1IS, TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable

The TLBI IPAS2LE1IS characteristics are:

## Purpose

If EL2 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 2 only translation table entry, from the final level of the translation table walk.
- SCR\_EL3.NS is 1.
- The entry would be used with the current VMID.
- The entry would be required to translate the specified IPA using the Non-secure EL1&0 translation regime.

The invalidation is not required to apply to caching structures that combine stage 1 and stage 2 translation table entries.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this instructions.

For more information about the architectural requirements for this instruction see 'Invalidation of TLB entries from stage 2 translations' in the ARMv8 ARM.

This System instruction is part of:

- The TLB maintenance instructions functional group.
- The Virtualization registers functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI IPAS2LE1IS is a 64-bit System instruction.

## Field descriptions

The TLBI IPAS2LE1IS 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	IPA[51:48]				IPA[47:12]			
IPA[47:12]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:40]

Reserved, RES0.

### IPA[51:48], bits [39:36]

In ARMv8.2:

Extension to IPA[47:12]. See IPA[47:12] for more details.

### In ARMv8.1 and ARMv8.0:

Reserved, RES0.

**IPA[47:12], bits [35:0]**

Bits[47:12] of the intermediate physical address to match.

When ARMv8.2-LPA is implemented, and 52-bit addresses and a 64KB translation granule are in use, IPA[51:48] form the upper part of the address value. Otherwise, IPA[51:48] are RES0.

**Executing the TLBI IPAS2LE1IS 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
IPAS2LE1IS	01	100	1000	0000	101

**Accessibility**

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
x	0	1	-	-	WO	WO
x	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, or SCR\_EL3.NS is 0, this instruction is a NOP.

# TLBI VAAE1, TLB Invalidate by VA, All ASID, EL1

The TLBI VAAE1 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, from any level of the translation table walk.
- The entry would be used to translate the specified address.
- 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.

### Note

For the EL1&0 translation regime, the invalidation applies to both:

- Global entries.
- Non-global entries with any ASID.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI VAAE1 is a 64-bit System instruction.

## Field descriptions

The TLBI VAAE1 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

### Bits [63:44]

Reserved, RES0.

### VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this operation, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:



- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

## Executing the TLBI VAAE1 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
VAAE1	01	000	1000	0111	011

## 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 VAAE1IS](#).

## 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.

# TLBI VAAE1IS, TLB Invalidate by VA, All ASID, EL1, Inner Shareable

The TLBI VAAE1IS 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, from any level of the translation table walk.
- The entry would be used to translate the specified address.
- 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.

---

### Note

For the EL1&0 translation regime, the invalidation applies to both:

- Global entries.
  - Non-global entries with any ASID.
- 

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI VAAE1IS is a 64-bit System instruction.

## Field descriptions

The TLBI VAAE1IS 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

### Bits [63:44]

Reserved, RES0.

### VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this operation, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

## Executing the TLBI VAAE1IS 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
VAAE1IS	01	000	1000	0011	011

## 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.

# TLBI VAALE1, TLB Invalidate by VA, All ASID, Last level, EL1

The TLBI VAALE1 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, from the final level of the translation table walk.
- The entry would be used to translate the specified address.
- 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.

### Note

For the EL1&0 translation regime, the invalidation applies to both:

- Global entries.
- Non-global entries with any ASID.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI VAALE1 is a 64-bit System instruction.

## Field descriptions

The TLBI VAALE1 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

### Bits [63:44]

Reserved, RES0.

### VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this operation, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

## Executing the TLBI VAALE1 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
VAALE1	01	000	1000	0111	111

## 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 VAALE1IS](#).

## 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.

# TLBI VAALE1IS, TLB Invalidate by VA, All ASID, EL1, Last Level, Inner Shareable

The TLBI VAALE1IS 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, from the final level of the translation table walk.
- The entry would be used to translate the specified address.
- 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.

### Note

For the EL1&0 translation regime, the invalidation applies to both:

- Global entries.
- Non-global entries with any ASID.

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI VAALE1IS is a 64-bit System instruction.

## Field descriptions

The TLBI VAALE1IS 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

### Bits [63:44]

Reserved, RES0.

### VA[55:12], bits [43:0]

Bits[55:12] of the virtual address to match. Any appropriate TLB entries that match the VA will be affected by this operation, regardless of the ASID.

If the TLB maintenance instructions are targeting a translation regime that is using AArch32, and so has a VA of only 32 bits, then the software must treat bits[55:32] as RES0.

The treatment of the low-order bits of this field depends on the translation granule size, as follows:

- Where a 4KB translation granule is being used, all bits are valid and used for the invalidation.
- Where a 16KB translation granule is being used, bits [1:0] of this field are RES0 and ignored when the instruction is executed, because VA[13:12] have no effect on the operation of the instruction.
- Where a 64KB translation granule is being used, bits [3:0] of this field are RES0 and ignored when the instruction is executed, because VA[15:12] have no effect on the operation of the instruction.

## Executing the TLBI VAALE1IS 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
VAALE1IS	01	000	1000	0011	111

## 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.

# 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<sup>®</sup> 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.

# 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.

# TLBI VAE2, TLB Invalidate by VA, EL2

The TLBI VAE2 characteristics are:

## Purpose

If EL2 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- The entry would be used to translate the specified address.
- SCR\_EL3.NS is 1.
- The entry would be required to translate using the EL2 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 VAE2 is a 64-bit System instruction.

## Field descriptions

The TLBI VAE2 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

### Bits [63:44]

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 VAE2 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
VAE2	01	100	1000	0111	001

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	-
0	0	1	-	-	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, or SCR\_EL3.NS is 0, this instruction is UNDEFINED.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI VAE2IS, TLB Invalidate by VA, EL2, Inner Shareable

The TLBI VAE2IS characteristics are:

## Purpose

If EL2 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- The entry would be used to translate the specified address.
- SCR\_EL3.NS is 1.
- The entry would be required to translate using the EL2 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 VAE2IS is a 64-bit System instruction.

## Field descriptions

The TLBI VAE2IS 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

### Bits [63:44]

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 VAE2IS 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
VAE2IS	01	100	1000	0011	001

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	-
0	0	1	-	-	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, or SCR\_EL3.NS is 0, this instruction is UNDEFINED.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI VAE3, TLB Invalidate by VA, EL3

The TLBI VAE3 characteristics are:

## Purpose

If EL3 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- The entry would be used to translate the specified address.
- The entry would be required to translate using the EL3 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 VAE3 is a 64-bit System instruction.

## Field descriptions

The TLBI VAE3 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

### Bits [63:44]

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 VAE3 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
VAE3	01	110	1000	0111	001

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	-	WO
0	1	1	-	n/a	-	WO
1	0	1	-	-	-	WO
1	1	1	-	n/a	-	WO

This table applies to all syntax that can be used to execute this instruction.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI VAE3IS, TLB Invalidate by VA, EL3, Inner Shareable

The TLBI VAE3IS characteristics are:

## Purpose

If EL3 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from any level of the translation table walk.
- The entry would be used to translate the specified address.
- The entry would be required to translate using the EL3 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 VAE3IS is a 64-bit System instruction.

## Field descriptions

The TLBI VAE3IS 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

### Bits [63:44]

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 VAE3IS 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
VAE3IS	01	110	1000	0011	001

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	-	WO
0	1	1	-	n/a	-	WO
1	0	1	-	-	-	WO
1	1	1	-	n/a	-	WO

This table applies to all syntax that can be used to execute this instruction.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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<sup>®</sup> 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.

# 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.

# TLBI VALE2, TLB Invalidate by VA, Last level, EL2

The TLBI VALE2 characteristics are:

## Purpose

If EL2 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from the final level of the translation table walk.
- The entry would be used to translate the specified address.
- SCR\_EL3.NS is 1.
- The entry would be required to translate using the EL2 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 VALE2 is a 64-bit System instruction.

## Field descriptions

The TLBI VALE2 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	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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	

### Bits [63:44]

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 VALE2 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
VALE2	01	100	1000	0111	101

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	-
0	0	1	-	-	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, or SCR\_EL3.NS is 0, this instruction is UNDEFINED.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI VALE2IS, TLB Invalidate by VA, Last level, EL2, Inner Shareable

The TLBI VALE2IS characteristics are:

## Purpose

If EL2 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from the final level of the translation table walk.
- The entry would be used to translate the specified address.
- SCR\_EL3.NS is 1.
- The entry would be required to translate using the EL2 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 VALE2IS is a 64-bit System instruction.

## Field descriptions

The TLBI VALE2IS 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

### Bits [63:44]

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 VALE2IS 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
VALE2IS	01	100	1000	0011	101

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	-
0	0	1	-	-	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If EL2 is not implemented, or SCR\_EL3.NS is 0, this instruction is UNDEFINED.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI VALE3, TLB Invalidate by VA, Last level, EL3

The TLBI VALE3 characteristics are:

## Purpose

If EL3 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from the final level of the translation table walk.
- The entry would be used to translate the specified address.
- The entry would be required to translate using the EL3 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 VALE3 is a 64-bit System instruction.

## Field descriptions

The TLBI VALE3 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				
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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				

### Bits [63:44]

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 VALE3 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
VALE3	01	110	1000	0111	101

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	-	WO
0	1	1	-	n/a	-	WO
1	0	1	-	-	-	WO
1	1	1	-	n/a	-	WO

This table applies to all syntax that can be used to execute this instruction.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI VALE3IS, TLB Invalidate by VA, Last level, EL3, Inner Shareable

The TLBI VALE3IS characteristics are:

## Purpose

If EL3 is implemented, invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 translation table entry, from the final level of the translation table walk.
- The entry would be used to translate the specified address.
- The entry would be required to translate using the EL3 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 VALE3IS is a 64-bit System instruction.

## Field descriptions

The TLBI VALE3IS 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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

### Bits [63:44]

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 VALE3IS 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
VALE3IS	01	110	1000	0011	101

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3
x	x	0	-	-	n/a	WO
0	0	1	-	-	-	WO
0	1	1	-	n/a	-	WO
1	0	1	-	-	-	WO
1	1	1	-	n/a	-	WO

This table applies to all syntax that can be used to execute this instruction.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI VMALLE1, TLB Invalidate by VMID, All at stage 1, EL1

The TLBI VMALLE1 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, from any level of the translation table walk.
- 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.

---

### Note

For the EL1&0 translation regime, the invalidation applies to both:

- Global entries.
  - Non-global entries with any ASID.
- 

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI VMALLE1 is a 64-bit System instruction.

## Field descriptions

TLBI VMALLE1 ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBI VMALLE1 instruction

This instruction is executed using TLBI with the following syntax:

TLBI <tlbi\_op>

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2	Rt
VMALLE1	01	000	1000	0111	000	1111

## 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 VMALLE1IS](#).

## Traps and enables

For a description of the prioritization of any generated exceptions, see section D1.13.2 (Synchronous exception prioritization) in the *ARM<sup>®</sup> 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI VMALLE1IS, TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable

The TLBI VMALLE1IS 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, from any level of the translation table walk.
- 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.

---

### Note

For the EL1&0 translation regime, the invalidation applies to both:

- Global entries.
  - Non-global entries with any ASID.
- 

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI VMALLE1IS is a 64-bit System instruction.

## Field descriptions

TLBI VMALLE1IS ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBI VMALLE1IS instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2	Rt
VMALLE1IS	01	000	1000	0011	000	11111

## 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<sup>®</sup> 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.

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI VMALLS12E1, TLB Invalidate by VMID, All at Stage 1 and 2, EL1

The TLBI VMALLS12E1 characteristics are:

## Purpose

Invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 or stage 2 translation table entry, from any level of the translation table walk.
- 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:
  - The entry would be required to translate an address using the Non-secure EL1&0 translation regime.
  - If EL2 is implemented, the entry would be used with the current VMID.

The invalidation only applies to the PE that executes this instruction.

---

### Note

For the EL1&0 translation regime, the invalidation applies to both:

- Global entries.
  - Non-global entries with any ASID.
- 

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI VMALLS12E1 is a 64-bit System instruction.

## Field descriptions

TLBI VMALLS12E1 ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBI VMALLS12E1 instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2	Rt
VMALLS12E1	01	100	1000	0111	110	11111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3

x	x	0	-	-	n/a	WO
0	0	1	-	-	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If [SCR\\_EL3.NS](#)=0, or EL2 is not implemented, this instruction executes as a [TLBI VMALLE1](#).

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# TLBI VMALLS12E1IS, TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner Shareable

The TLBI VMALLS12E1IS characteristics are:

## Purpose

Invalidate cached copies of translation table entries from TLBs that meet all the following requirements:

- The entry is a stage 1 or stage 2 translation table entry, from any level of the translation table walk.
- 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:
  - The entry would be required to translate an address using the Non-secure EL1&0 translation regime.
  - If EL2 is implemented, the entry would be used with the current VMID.

The invalidation applies to all PEs in the same Inner Shareable shareability domain as the PE that executes this instructions.

---

### Note

For the EL1&0 translation regime, the invalidation applies to both:

- Global entries.
  - Non-global entries with any ASID.
- 

This System instruction is part of the TLB maintenance instructions functional group.

## Configuration

There are no configuration notes.

## Attributes

TLBI VMALLS12E1IS is a 64-bit System instruction.

## Field descriptions

TLBI VMALLS12E1IS ignores the value in the register specified by the instruction encoding. Software does not have to write a value to the register before issuing this instruction.

## Executing the TLBI VMALLS12E1IS instruction

This instruction is executed using TLBI with the following syntax:

```
TLBI <tlbi_op>
```

This syntax uses the following encoding in the System instruction encoding space:

<tlbi_op>	op0	op1	CRn	CRm	op2	Rt
VMALLS12E1IS	01	100	1000	0011	110	11111

## Accessibility

The instruction is executable as follows:

Control			Accessibility			
E2H	TGE	NS	EL0	EL1	EL2	EL3

x	x	0	-	-	n/a	WO
0	0	1	-	-	WO	WO
0	1	1	-	n/a	WO	WO
1	0	1	-	-	WO	WO
1	1	1	-	n/a	WO	WO

This table applies to all syntax that can be used to execute this instruction.

If [SCR\\_EL3.NS](#)=0, or EL2 is not implemented, this instruction executes as a [TLBI VMALLE1IS](#).

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# System Register index by instruction and encoding

Below are indexes for registers and operations accessed in the following ways:

For AArch32

- [MRC/MCR](#)
- [MRS/MSR](#)
- [VMRS/VMSR](#)
- [MRRC/MCRR](#)

For AArch64

- [MRS/MSR](#)
- [TLBI](#)
- [SYSL/SYS](#)
- [DC/IC](#)
- [AT](#)

## Registers and operations in AArch32

### Accessed using MRC/MCR:

coproc	Register selectors			opc2	Name	Description
	opc1	CRn	CRm			
1110	000	0000	0000	000	<a href="#">DBGDIDR</a>	Debug ID Register
1110	000	0000	0000	010	<a href="#">DBGDTRRXext</a>	Debug OS Lock Data Transfer Register, Receive, External View
1110	000	0000	0001	000	<a href="#">DBGDSCRint</a>	Debug Status and Control Register, Internal View
1110	000	0000	0010	000	<a href="#">DBGDCCINT</a>	DCC Interrupt Enable Register
1110	000	0000	0010	010	<a href="#">DBGDSCRext</a>	Debug Status and Control Register, External View
1110	000	0000	0011	010	<a href="#">DBGDTRTXext</a>	Debug OS Lock Data Transfer Register, Transmit
1110	000	0000	0101	000	<a href="#">DBGDTRRXint</a>	Debug Data Transfer Register, Receive
1110	000	0000	0101	000	<a href="#">DBGDTRTXint</a>	Debug Data Transfer Register, Transmit
1110	000	0000	0110	000	<a href="#">DBGWFAR</a>	Debug Watchpoint Fault Address Register
1110	000	0000	0110	010	<a href="#">DBGOSECCR</a>	Debug OS Lock Exception Catch Control Register
1110	000	0000	0111	000	<a href="#">DBGVCR</a>	Debug Vector Catch Register
1110	000	0000	xxxx	100	<a href="#">DBGBVR&lt;n&gt;</a>	Debug Breakpoint Value Registers
1110	000	0000	xxxx	101	<a href="#">DBGBCR&lt;n&gt;</a>	Debug Breakpoint Control Registers
1110	000	0000	xxxx	110	<a href="#">DBGWVR&lt;n&gt;</a>	Debug Watchpoint Value Registers
1110	000	0000	xxxx	111	<a href="#">DBGWCR&lt;n&gt;</a>	Debug Watchpoint Control Registers
1110	000	0001	0000	000	<a href="#">DBGDRAR</a>	Debug ROM Address Register
1110	000	0001	0000	100	<a href="#">DBGOSLAR</a>	Debug OS Lock Access Register
1110	000	0001	0001	100	<a href="#">DBGOSLSR</a>	Debug OS Lock Status Register
1110	000	0001	0011	100	<a href="#">DBGOSDLR</a>	Debug OS Double Lock Register
1110	000	0001	0100	100	<a href="#">DBGPRCR</a>	Debug Power Control Register
1110	000	0001	xxxx	001	<a href="#">DBGBXVR&lt;n&gt;</a>	Debug Breakpoint Extended Value Registers
1110	000	0010	0000	000	<a href="#">DBGDSAR</a>	Debug Self Address Register
1110	000	0111	0000	111	<a href="#">DBGDEVID2</a>	Debug Device ID register 2
1110	000	0111	0001	111	<a href="#">DBGDEVID1</a>	Debug Device ID register 1
1110	000	0111	0010	111	<a href="#">DBGDEVID</a>	Debug Device ID register 0
1110	000	0111	1000	110	<a href="#">DBGCLAIMSET</a>	Debug Claim Tag Set register
1110	000	0111	1001	110	<a href="#">DBGCLAIMCLR</a>	Debug Claim Tag Clear register



Register selectors					Name	Description
coproc	opc1	CRn	CRm	opc2		
1110	000	0111	1110	110	<a href="#">DBGAUTHSTATUS</a>	Debug Authentication Status register
1110	111	0000	0000	000	<a href="#">JIDR</a>	Jazelle ID Register
1110	111	0001	0000	000	<a href="#">JOSCR</a>	Jazelle OS Control Register
1110	111	0010	0000	000	<a href="#">JMCR</a>	Jazelle Main Configuration Register
1111	000	0000	0000	000	<a href="#">MIDR</a>	Main ID Register
1111	000	0000	0000	001	<a href="#">CTR</a>	Cache Type Register
1111	000	0000	0000	010	<a href="#">TCMTR</a>	TCM Type Register
1111	000	0000	0000	011	<a href="#">TLBTR</a>	TLB Type Register
1111	000	0000	0000	101	<a href="#">MPIDR</a>	Multiprocessor Affinity Register
1111	000	0000	0000	110	<a href="#">REVIDR</a>	Revision ID Register
1111	000	0000	0001	000	<a href="#">ID_PFR0</a>	Processor Feature Register 0
1111	000	0000	0001	001	<a href="#">ID_PFR1</a>	Processor Feature Register 1
1111	000	0000	0001	010	<a href="#">ID_DFR0</a>	Debug Feature Register 0
1111	000	0000	0001	011	<a href="#">ID_AFR0</a>	Auxiliary Feature Register 0
1111	000	0000	0001	100	<a href="#">ID_MMFR0</a>	Memory Model Feature Register 0
1111	000	0000	0001	101	<a href="#">ID_MMFR1</a>	Memory Model Feature Register 1
1111	000	0000	0001	110	<a href="#">ID_MMFR2</a>	Memory Model Feature Register 2
1111	000	0000	0001	111	<a href="#">ID_MMFR3</a>	Memory Model Feature Register 3
1111	000	0000	0010	000	<a href="#">ID_ISAR0</a>	Instruction Set Attribute Register 0
1111	000	0000	0010	001	<a href="#">ID_ISAR1</a>	Instruction Set Attribute Register 1
1111	000	0000	0010	010	<a href="#">ID_ISAR2</a>	Instruction Set Attribute Register 2
1111	000	0000	0010	011	<a href="#">ID_ISAR3</a>	Instruction Set Attribute Register 3
1111	000	0000	0010	100	<a href="#">ID_ISAR4</a>	Instruction Set Attribute Register 4
1111	000	0000	0010	101	<a href="#">ID_ISAR5</a>	Instruction Set Attribute Register 5
1111	000	0000	0010	110	<a href="#">ID_MMFR4</a>	Memory Model Feature Register 4
1111	000	0001	0000	000	<a href="#">SCTLR</a>	System Control Register
1111	000	0001	0000	001	<a href="#">ACTLR</a>	Auxiliary Control Register
1111	000	0001	0000	010	<a href="#">CPACR</a>	Architectural Feature Access Control Register
1111	000	0001	0000	011	<a href="#">ACTLR2</a>	Auxiliary Control Register 2
1111	000	0001	0001	000	<a href="#">SCR</a>	Secure Configuration Register
1111	000	0001	0001	001	<a href="#">SDER</a>	Secure Debug Enable Register
1111	000	0001	0001	010	<a href="#">NSACR</a>	Non-Secure Access Control Register
1111	000	0001	0011	001	<a href="#">SDCR</a>	Secure Debug Control Register
1111	000	0010	0000	000	<a href="#">TTBR0</a>	Translation Table Base Register 0
1111	000	0010	0000	001	<a href="#">TTBR1</a>	Translation Table Base Register 1
1111	000	0010	0000	010	<a href="#">TTBCR</a>	Translation Table Base Control Register
1111	000	0010	0000	011	<a href="#">TTBCR2</a>	Translation Table Base Control Register 2
1111	000	0011	0000	000	<a href="#">DACR</a>	Domain Access Control Register
1111	000	0100	0110	000	<a href="#">ICC_PMR</a>	Interrupt Controller Interrupt Priority Mask Register
1111	000	0100	0110	000	<a href="#">ICV_PMR</a>	Interrupt Controller Virtual Interrupt Priority Mask Register
1111	000	0101	0000	000	<a href="#">DFSR</a>	Data Fault Status Register
1111	000	0101	0000	001	<a href="#">IFSR</a>	Instruction Fault Status Register
1111	000	0101	0001	000	<a href="#">ADFSR</a>	Auxiliary Data Fault Status Register
1111	000	0101	0001	001	<a href="#">AIFSR</a>	Auxiliary Instruction Fault Status Register
1111	000	0110	0000	000	<a href="#">DFAR</a>	Data Fault Address Register
1111	000	0110	0000	010	<a href="#">IFAR</a>	Instruction Fault Address Register
1111	000	0111	0001	000	<a href="#">ICIALLUIS</a>	Instruction Cache Invalidate All to PoU, Inner Shareable

coproc	Register selectors			opc2	Name	Description
	opc1	CRn	CRm			
1111	000	0111	0001	110	<a href="#">BPIALLIS</a>	Branch Predictor Invalidate All, Inner Shareable
1111	000	0111	0100	000	<a href="#">PAR</a>	Physical Address Register
1111	000	0111	0101	000	<a href="#">ICIALLU</a>	Instruction Cache Invalidate All to PoU
1111	000	0111	0101	001	<a href="#">ICIMVAU</a>	Instruction Cache line Invalidate by VA to PoU
1111	000	0111	0101	100	<a href="#">CPI5ISB</a>	Instruction Synchronization Barrier System instruction
1111	000	0111	0101	110	<a href="#">BPIALL</a>	Branch Predictor Invalidate All
1111	000	0111	0101	111	<a href="#">BPIMVA</a>	Branch Predictor Invalidate by VA
1111	000	0111	0110	001	<a href="#">DCIMVAC</a>	Data Cache line Invalidate by VA to PoC
1111	000	0111	0110	010	<a href="#">DCISW</a>	Data Cache line Invalidate by Set/Way
1111	000	0111	1000	000	<a href="#">ATS1CPR</a>	Address Translate Stage 1 Current state PL1 Read
1111	000	0111	1000	001	<a href="#">ATS1CPW</a>	Address Translate Stage 1 Current state PL1 Write
1111	000	0111	1000	010	<a href="#">ATS1CUR</a>	Address Translate Stage 1 Current state Unprivileged Read
1111	000	0111	1000	011	<a href="#">ATS1CUW</a>	Address Translate Stage 1 Current state Unprivileged Write
1111	000	0111	1000	100	<a href="#">ATS12NSOPR</a>	Address Translate Stages 1 and 2 Non-secure Only PL1 Read
1111	000	0111	1000	101	<a href="#">ATS12NSOPW</a>	Address Translate Stages 1 and 2 Non-secure Only PL1 Write
1111	000	0111	1000	110	<a href="#">ATS12NSOUR</a>	Address Translate Stages 1 and 2 Non-secure Only Unprivileged Read
1111	000	0111	1000	111	<a href="#">ATS12NSOUW</a>	Address Translate Stages 1 and 2 Non-secure Only Unprivileged Write
1111	000	0111	1001	000	<a href="#">ATS1CPRP</a>	Address Translate Stage 1 Current state PL1 Read PAN
1111	000	0111	1001	001	<a href="#">ATS1CPWP</a>	Address Translate Stage 1 Current state PL1 Write PAN
1111	000	0111	1010	001	<a href="#">DCCMVAC</a>	Data Cache line Clean by VA to PoC
1111	000	0111	1010	010	<a href="#">DCCSW</a>	Data Cache line Clean by Set/Way
1111	000	0111	1010	100	<a href="#">CPI5DSB</a>	Data Synchronization Barrier System instruction
1111	000	0111	1010	101	<a href="#">CPI5DMB</a>	Data Memory Barrier System instruction
1111	000	0111	1011	001	<a href="#">DCCMVAU</a>	Data Cache line Clean by VA to PoU
1111	000	0111	1110	001	<a href="#">DCCIMVAC</a>	Data Cache line Clean and Invalidate by VA to PoC
1111	000	0111	1110	010	<a href="#">DCCISW</a>	Data Cache line Clean and Invalidate by Set/Way
1111	000	1000	0011	000	<a href="#">TLBIALLIS</a>	TLB Invalidate All, Inner Shareable
1111	000	1000	0011	001	<a href="#">TLBIMVAIS</a>	TLB Invalidate by VA, Inner Shareable
1111	000	1000	0011	010	<a href="#">TLBIASIDIS</a>	TLB Invalidate by ASID match, Inner Shareable
1111	000	1000	0011	011	<a href="#">TLBIMVAAIS</a>	TLB Invalidate by VA, All ASID, Inner Shareable
1111	000	1000	0011	101	<a href="#">TLBIMVALIS</a>	TLB Invalidate by VA, Last level, Inner Shareable
1111	000	1000	0011	111	<a href="#">TLBIMVAALIS</a>	TLB Invalidate by VA, All ASID, Last level, Inner Shareable
1111	000	1000	0101	000	<a href="#">ITLBIALL</a>	Instruction TLB Invalidate All
1111	000	1000	0101	001	<a href="#">ITLBIMVA</a>	Instruction TLB Invalidate by VA
1111	000	1000	0101	010	<a href="#">ITLBIASID</a>	Instruction TLB Invalidate by ASID match
1111	000	1000	0110	000	<a href="#">DTLBIALL</a>	Data TLB Invalidate All
1111	000	1000	0110	001	<a href="#">DTLBIMVA</a>	Data TLB Invalidate by VA
1111	000	1000	0110	010	<a href="#">DTLBIASID</a>	Data TLB Invalidate by ASID match
1111	000	1000	0111	000	<a href="#">TLBIALL</a>	TLB Invalidate All
1111	000	1000	0111	001	<a href="#">TLBIMVA</a>	TLB Invalidate by VA
1111	000	1000	0111	010	<a href="#">TLBIASID</a>	TLB Invalidate by ASID match
1111	000	1000	0111	011	<a href="#">TLBIMVAA</a>	TLB Invalidate by VA, All ASID

coproc	Register selectors			opc2	Name	Description
	opc1	CRn	CRm			
1111	000	1000	0111	101	<a href="#">TLBIMVAL</a>	TLB Invalidate by VA, Last level
1111	000	1000	0111	111	<a href="#">TLBIMVAAL</a>	TLB Invalidate by VA, All ASID, Last level
1111	000	1001	1100	000	<a href="#">PMCR</a>	Performance Monitors Control Register
1111	000	1001	1100	001	<a href="#">PMCNTENSET</a>	Performance Monitors Count Enable Set register
1111	000	1001	1100	010	<a href="#">PMCNTENCLR</a>	Performance Monitors Count Enable Clear register
1111	000	1001	1100	011	<a href="#">PMOVSr</a>	Performance Monitors Overflow Flag Status Register
1111	000	1001	1100	100	<a href="#">PMSWINC</a>	Performance Monitors Software Increment register
1111	000	1001	1100	101	<a href="#">PMSELR</a>	Performance Monitors Event Counter Selection Register
1111	000	1001	1100	110	<a href="#">PMCEID0</a>	Performance Monitors Common Event Identification register 0
1111	000	1001	1100	111	<a href="#">PMCEID1</a>	Performance Monitors Common Event Identification register 1
1111	000	1001	1101	000	<a href="#">PMCCNTR</a>	Performance Monitors Cycle Count Register
1111	000	1001	1101	001	<a href="#">PMXEVTYPER</a>	Performance Monitors Selected Event Type Register
1111	000	1001	1101	010	<a href="#">PMXEVCNTR</a>	Performance Monitors Selected Event Count Register
1111	000	1001	1110	000	<a href="#">PMUSERENR</a>	Performance Monitors User Enable Register
1111	000	1001	1110	001	<a href="#">PMINTENSET</a>	Performance Monitors Interrupt Enable Set register
1111	000	1001	1110	010	<a href="#">PMINTENCLR</a>	Performance Monitors Interrupt Enable Clear register
1111	000	1001	1110	011	<a href="#">PMOVSSET</a>	Performance Monitors Overflow Flag Status Set register
1111	000	1001	1110	100	<a href="#">PMCEID2</a>	Performance Monitors Common Event Identification register 2
1111	000	1001	1110	101	<a href="#">PMCEID3</a>	Performance Monitors Common Event Identification register 3
1111	000	1010	0010	000	<a href="#">PRRR</a>	Primary Region Remap Register
1111	000	1010	0010	000	<a href="#">MAIR0</a>	Memory Attribute Indirection Register 0
1111	000	1010	0010	001	<a href="#">NMRR</a>	Normal Memory Remap Register
1111	000	1010	0010	001	<a href="#">MAIR1</a>	Memory Attribute Indirection Register 1
1111	000	1010	0011	000	<a href="#">AMAIRO</a>	Auxiliary Memory Attribute Indirection Register 0
1111	000	1010	0011	001	<a href="#">AMAIR1</a>	Auxiliary Memory Attribute Indirection Register 1
1111	000	1100	0000	000	<a href="#">VBAR</a>	Vector Base Address Register
1111	000	1100	0000	001	<a href="#">MVBAR</a>	Monitor Vector Base Address Register
1111	000	1100	0000	001	<a href="#">RVBAR</a>	Reset Vector Base Address Register
1111	000	1100	0000	010	<a href="#">RMR</a>	Reset Management Register
1111	000	1100	0001	000	<a href="#">ISR</a>	Interrupt Status Register
1111	000	1100	1000	000	<a href="#">ICC_IAR0</a>	Interrupt Controller Interrupt Acknowledge Register 0
1111	000	1100	1000	000	<a href="#">ICV_IAR0</a>	Interrupt Controller Virtual Interrupt Acknowledge Register 0
1111	000	1100	1000	001	<a href="#">ICC_EOIR0</a>	Interrupt Controller End Of Interrupt Register 0
1111	000	1100	1000	001	<a href="#">ICV_EOIR0</a>	Interrupt Controller Virtual End Of Interrupt Register 0
1111	000	1100	1000	010	<a href="#">ICC_HPPIR0</a>	Interrupt Controller Highest Priority Pending Interrupt Register 0
1111	000	1100	1000	010	<a href="#">ICV_HPPIR0</a>	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
1111	000	1100	1000	011	<a href="#">ICC_BPR0</a>	Interrupt Controller Binary Point Register 0
1111	000	1100	1000	011	<a href="#">ICV_BPR0</a>	Interrupt Controller Virtual Binary Point Register 0
1111	000	1100	1000	1xx	<a href="#">ICC_AP0R&lt;n&gt;</a>	Interrupt Controller Active Priorities Group 0 Registers

coproc	Register selectors			opc2	Name	Description
	opc1	CRn	CRm			
1111	000	1100	1000	1xx	<a href="#">ICV_AP0R&lt;n&gt;</a>	Interrupt Controller Virtual Active Priorities Group 0 Registers
1111	000	1100	1001	0xx	<a href="#">ICC_APIR&lt;n&gt;</a>	Interrupt Controller Active Priorities Group 1 Registers
1111	000	1100	1001	0xx	<a href="#">ICV_APIR&lt;n&gt;</a>	Interrupt Controller Virtual Active Priorities Group 1 Registers
1111	000	1100	1011	001	<a href="#">ICC_DIR</a>	Interrupt Controller Deactivate Interrupt Register
1111	000	1100	1011	001	<a href="#">ICV_DIR</a>	Interrupt Controller Deactivate Virtual Interrupt Register
1111	000	1100	1011	011	<a href="#">ICC_RPR</a>	Interrupt Controller Running Priority Register
1111	000	1100	1011	011	<a href="#">ICV_RPR</a>	Interrupt Controller Virtual Running Priority Register
1111	000	1100	1100	000	<a href="#">ICC_IAR1</a>	Interrupt Controller Interrupt Acknowledge Register 1
1111	000	1100	1100	000	<a href="#">ICV_IAR1</a>	Interrupt Controller Virtual Interrupt Acknowledge Register 1
1111	000	1100	1100	001	<a href="#">ICC_EOIR1</a>	Interrupt Controller End Of Interrupt Register 1
1111	000	1100	1100	001	<a href="#">ICV_EOIR1</a>	Interrupt Controller Virtual End Of Interrupt Register 1
1111	000	1100	1100	010	<a href="#">ICC_HPIR1</a>	Interrupt Controller Highest Priority Pending Interrupt Register 1
1111	000	1100	1100	010	<a href="#">ICV_HPIR1</a>	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
1111	000	1100	1100	011	<a href="#">ICC_BPR1</a>	Interrupt Controller Binary Point Register 1
1111	000	1100	1100	011	<a href="#">ICV_BPR1</a>	Interrupt Controller Virtual Binary Point Register 1
1111	000	1100	1100	100	<a href="#">ICC_CTLR</a>	Interrupt Controller Control Register
1111	000	1100	1100	100	<a href="#">ICV_CTLR</a>	Interrupt Controller Virtual Control Register
1111	000	1100	1100	101	<a href="#">ICC_SRE</a>	Interrupt Controller System Register Enable register
1111	000	1100	1100	110	<a href="#">ICC_IGRPEN0</a>	Interrupt Controller Interrupt Group 0 Enable register
1111	000	1100	1100	110	<a href="#">ICV_IGRPEN0</a>	Interrupt Controller Virtual Interrupt Group 0 Enable register
1111	000	1100	1100	111	<a href="#">ICC_IGRPEN1</a>	Interrupt Controller Interrupt Group 1 Enable register
1111	000	1100	1100	111	<a href="#">ICV_IGRPEN1</a>	Interrupt Controller Virtual Interrupt Group 1 Enable register
1111	000	1101	0000	000	<a href="#">FCSEIDR</a>	FCSE Process ID register
1111	000	1101	0000	001	<a href="#">CONTEXTIDR</a>	Context ID Register
1111	000	1101	0000	010	<a href="#">TPIDRURW</a>	PL0 Read/Write Software Thread ID Register
1111	000	1101	0000	011	<a href="#">TPIDRURO</a>	PL0 Read-Only Software Thread ID Register
1111	000	1101	0000	100	<a href="#">TPIDRPRW</a>	PL1 Software Thread ID Register
1111	000	1110	0000	000	<a href="#">CNTFRQ</a>	Counter-timer Frequency register
1111	000	1110	0001	000	<a href="#">CNTKCTL</a>	Counter-timer Kernel Control register
1111	000	1110	0010	000	<a href="#">CNTP_TVAL</a>	Counter-timer Physical Timer TimerValue register
1111	000	1110	0010	001	<a href="#">CNTP_CTL</a>	Counter-timer Physical Timer Control register
1111	000	1110	0011	000	<a href="#">CNTV_TVAL</a>	Counter-timer Virtual Timer TimerValue register
1111	000	1110	0011	001	<a href="#">CNTV_CTL</a>	Counter-timer Virtual Timer Control register
1111	000	1110	10xx	xxx	<a href="#">PMEVCNTR&lt;n&gt;</a>	Performance Monitors Event Count Registers
1111	000	1110	1111	111	<a href="#">PMCCFILTR</a>	Performance Monitors Cycle Count Filter Register
1111	000	1110	11xx	xxx	<a href="#">PMEVTYPEPER&lt;n&gt;</a>	Performance Monitors Event Type Registers
1111	001	0000	0000	000	<a href="#">CCSIDR</a>	Current Cache Size ID Register
1111	001	0000	0000	001	<a href="#">CLIDR</a>	Cache Level ID Register
1111	001	0000	0000	111	<a href="#">AIDR</a>	Auxiliary ID Register

coproc	Register selectors			opc2	Name	Description
	opc1	CRn	CRm			
1111	010	0000	0000	000	<a href="#">CSSELR</a>	Cache Size Selection Register
1111	011	0100	0101	000	<a href="#">DPSR</a>	Debug Saved Program Status Register
1111	011	0100	0101	001	<a href="#">DLR</a>	Debug Link Register
1111	100	0000	0000	000	<a href="#">VPIDR</a>	Virtualization Processor ID Register
1111	100	0000	0000	101	<a href="#">VMPIDR</a>	Virtualization Multiprocessor ID Register
1111	100	0001	0000	000	<a href="#">HSCTLR</a>	Hyp System Control Register
1111	100	0001	0000	001	<a href="#">HACTLR</a>	Hyp Auxiliary Control Register
1111	100	0001	0000	011	<a href="#">HACTLR2</a>	Hyp Auxiliary Control Register 2
1111	100	0001	0001	000	<a href="#">HCR</a>	Hyp Configuration Register
1111	100	0001	0001	001	<a href="#">HDCR</a>	Hyp Debug Control Register
1111	100	0001	0001	010	<a href="#">HCPTR</a>	Hyp Architectural Feature Trap Register
1111	100	0001	0001	011	<a href="#">HSTR</a>	Hyp System Trap Register
1111	100	0001	0001	100	<a href="#">HCR2</a>	Hyp Configuration Register 2
1111	100	0001	0001	111	<a href="#">HACR</a>	Hyp Auxiliary Configuration Register
1111	100	0010	0000	010	<a href="#">HTCR</a>	Hyp Translation Control Register
1111	100	0010	0001	010	<a href="#">VTCR</a>	Virtualization Translation Control Register
1111	100	0101	0001	000	<a href="#">HADEFSR</a>	Hyp Auxiliary Data Fault Status Register
1111	100	0101	0001	001	<a href="#">HAIFSR</a>	Hyp Auxiliary Instruction Fault Status Register
1111	100	0101	0010	000	<a href="#">HSR</a>	Hyp Syndrome Register
1111	100	0110	0000	000	<a href="#">HDFAR</a>	Hyp Data Fault Address Register
1111	100	0110	0000	010	<a href="#">HIFAR</a>	Hyp Instruction Fault Address Register
1111	100	0110	0000	100	<a href="#">HPFAR</a>	Hyp IPA Fault Address Register
1111	100	0111	1000	000	<a href="#">ATS1HR</a>	Address Translate Stage 1 Hyp mode Read
1111	100	0111	1000	001	<a href="#">ATS1HW</a>	Address Translate Stage 1 Hyp mode Write
1111	100	1000	0000	001	<a href="#">TLBIIPAS2IS</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Inner Shareable
1111	100	1000	0000	101	<a href="#">TLBIIPAS2LIS</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, Inner Shareable
1111	100	1000	0011	000	<a href="#">TLBIALLHIS</a>	TLB Invalidate All, Hyp mode, Inner Shareable
1111	100	1000	0011	001	<a href="#">TLBIMVAHIS</a>	TLB Invalidate by VA, Hyp mode, Inner Shareable
1111	100	1000	0011	100	<a href="#">TLBIALLNSNHIS</a>	TLB Invalidate All, Non-Secure Non-Hyp, Inner Shareable
1111	100	1000	0011	101	<a href="#">TLBIMVALHIS</a>	TLB Invalidate by VA, Last level, Hyp mode, Inner Shareable
1111	100	1000	0100	001	<a href="#">TLBIIPAS2</a>	TLB Invalidate by Intermediate Physical Address, Stage 2
1111	100	1000	0100	101	<a href="#">TLBIIPAS2L</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level
1111	100	1000	0111	000	<a href="#">TLBIALLH</a>	TLB Invalidate All, Hyp mode
1111	100	1000	0111	001	<a href="#">TLBIMVAH</a>	TLB Invalidate by VA, Hyp mode
1111	100	1000	0111	100	<a href="#">TLBIALLNSNH</a>	TLB Invalidate All, Non-Secure Non-Hyp
1111	100	1000	0111	101	<a href="#">TLBIMVALH</a>	TLB Invalidate by VA, Last level, Hyp mode
1111	100	1010	0010	000	<a href="#">HMAIR0</a>	Hyp Memory Attribute Indirection Register 0
1111	100	1010	0010	001	<a href="#">HMAIR1</a>	Hyp Memory Attribute Indirection Register 1
1111	100	1010	0011	000	<a href="#">HAMAIR0</a>	Hyp Auxiliary Memory Attribute Indirection Register 0
1111	100	1010	0011	001	<a href="#">HAMAIR1</a>	Hyp Auxiliary Memory Attribute Indirection Register 1
1111	100	1100	0000	000	<a href="#">HVBAR</a>	Hyp Vector Base Address Register
1111	100	1100	0000	010	<a href="#">HRMR</a>	Hyp Reset Management Register
1111	100	1100	1000	0xx	<a href="#">ICH_AP0R&lt;n&gt;</a>	Interrupt Controller Hyp Active Priorities Group 0 Registers

Register selectors					Name	Description
coproc	opc1	CRn	CRm	opc2		
1111	100	1100	1001	0xx	<a href="#">ICH_APIR&lt;n&gt;</a>	Interrupt Controller Hyp Active Priorities Group 1 Registers
1111	100	1100	1001	101	<a href="#">ICC_HSRE</a>	Interrupt Controller Hyp System Register Enable register
1111	100	1100	1011	000	<a href="#">ICH_HCR</a>	Interrupt Controller Hyp Control Register
1111	100	1100	1011	001	<a href="#">ICH_VTR</a>	Interrupt Controller VGIC Type Register
1111	100	1100	1011	010	<a href="#">ICH_MISR</a>	Interrupt Controller Maintenance Interrupt State Register
1111	100	1100	1011	011	<a href="#">ICH_EISR</a>	Interrupt Controller End of Interrupt Status Register
1111	100	1100	1011	101	<a href="#">ICH_ELRSR</a>	Interrupt Controller Empty List Register Status Register
1111	100	1100	1011	111	<a href="#">ICH_VMCR</a>	Interrupt Controller Virtual Machine Control Register
1111	100	1100	110x	xxx	<a href="#">ICH_LR&lt;n&gt;</a>	Interrupt Controller List Registers
1111	100	1100	111x	xxx	<a href="#">ICH_LRC&lt;n&gt;</a>	Interrupt Controller List Registers
1111	100	1101	0000	010	<a href="#">HTPIDR</a>	Hyp Software Thread ID Register
1111	100	1110	0001	000	<a href="#">CNTHCTL</a>	Counter-timer Hyp Control register
1111	100	1110	0010	000	<a href="#">CNTHP_TVAL</a>	Counter-timer Hyp Physical Timer TimerValue register
1111	100	1110	0010	001	<a href="#">CNTHP_CTL</a>	Counter-timer Hyp Physical Timer Control register
1111	110	1100	1100	100	<a href="#">ICC_MCTLR</a>	Interrupt Controller Monitor Control Register
1111	110	1100	1100	101	<a href="#">ICC_MSRE</a>	Interrupt Controller Monitor System Register Enable register
1111	110	1100	1100	111	<a href="#">ICC_MGRPEN1</a>	Interrupt Controller Monitor Interrupt Group 1 Enable register

## Accessed using MRS/MSR:

Register selectors			Name	Description
m	m1	R		
0	1110	1	<a href="#">SPSR_fiq</a>	Saved Program Status Register (FIQ mode)
1	0000	1	<a href="#">SPSR_irq</a>	Saved Program Status Register (IRQ mode)
1	0010	1	<a href="#">SPSR_svc</a>	Saved Program Status Register (Supervisor mode)
1	0100	1	<a href="#">SPSR_abt</a>	Saved Program Status Register (Abort mode)
1	0110	1	<a href="#">SPSR_und</a>	Saved Program Status Register (Undefined mode)
1	1100	1	<a href="#">SPSR_mon</a>	Saved Program Status Register (Monitor mode)
1	1110	0	<a href="#">ELR_hyp</a>	Exception Link Register (Hyp mode)
1	1110	1	<a href="#">SPSR_hyp</a>	Saved Program Status Register (Hyp mode)

## Accessed using VMRS/VMSR:

Register selectors	Name	Description
spec_reg		
0000	<a href="#">FPSID</a>	Floating-Point System ID register
0001	<a href="#">FPSCR</a>	Floating-Point Status and Control Register
0101	<a href="#">MVFR2</a>	Media and VFP Feature Register 2
0110	<a href="#">MVFR1</a>	Media and VFP Feature Register 1
0111	<a href="#">MVFR0</a>	Media and VFP Feature Register 0
1000	<a href="#">FPEXC</a>	Floating-Point Exception Control register



## Accessed using MRRC/MCRR:

Register selectors			Name	Description
coproc	opc1	CRm		
1110	0000	0001	<a href="#">DBGDRAR</a>	Debug ROM Address Register
1110	0000	0010	<a href="#">DBGDSAR</a>	Debug Self Address Register
1111	0000	0010	<a href="#">TTBR0</a>	Translation Table Base Register 0
1111	0001	0010	<a href="#">TTBR1</a>	Translation Table Base Register 1
1111	0100	0010	<a href="#">HTTBR</a>	Hyp Translation Table Base Register
1111	0110	0010	<a href="#">VTTBR</a>	Virtualization Translation Table Base Register
1111	0000	0111	<a href="#">PAR</a>	Physical Address Register
1111	0000	1001	<a href="#">PMCCNTR</a>	Performance Monitors Cycle Count Register
1111	0000	1100	<a href="#">ICC_SGI1R</a>	Interrupt Controller Software Generated Interrupt Group 1 Register
1111	0001	1100	<a href="#">ICC_ASGI1R</a>	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
1111	0010	1100	<a href="#">ICC_SGI0R</a>	Interrupt Controller Software Generated Interrupt Group 0 Register
1111	0000	1110	<a href="#">CNTPCT</a>	Counter-timer Physical Count register
1111	0001	1110	<a href="#">CNTVCT</a>	Counter-timer Virtual Count register
1111	0010	1110	<a href="#">CNTP_CVAL</a>	Counter-timer Physical Timer CompareValue register
1111	0011	1110	<a href="#">CNTV_CVAL</a>	Counter-timer Virtual Timer CompareValue register
1111	0100	1110	<a href="#">CNTVOFF</a>	Counter-timer Virtual Offset register
1111	0110	1110	<a href="#">CNTHP_CVAL</a>	Counter-timer Hyp Physical CompareValue register

## Registers and operations in AArch64

### Accessed using MRS/MSR:

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
10	000	0000	0000	010	<a href="#">OSDTRRX_EL1</a>	OS Lock Data Transfer Register, Receive
10	011	0000	0001	000	<a href="#">MDCCSR_EL0</a>	Monitor DCC Status Register
10	000	0000	0010	000	<a href="#">MDCCINT_EL1</a>	Monitor DCC Interrupt Enable Register
10	000	0000	0010	010	<a href="#">MDSCR_EL1</a>	Monitor Debug System Control Register
10	000	0000	0011	010	<a href="#">OSDTRTX_EL1</a>	OS Lock Data Transfer Register, Transmit
10	011	0000	0100	000	<a href="#">DBGDTR_EL0</a>	Debug Data Transfer Register, half-duplex
10	011	0000	0101	000	<a href="#">DBGDTRRX_EL0</a>	Debug Data Transfer Register, Receive
10	011	0000	0101	000	<a href="#">DBGDTRTX_EL0</a>	Debug Data Transfer Register, Transmit
10	000	0000	0110	010	<a href="#">OSECCR_EL1</a>	OS Lock Exception Catch Control Register
10	100	0000	0111	000	<a href="#">DBGVCR32_EL2</a>	Debug Vector Catch Register
10	000	0000	xxxx	100	<a href="#">DBGBVR&lt;n&gt;_EL1</a>	Debug Breakpoint Value Registers
10	000	0000	xxxx	101	<a href="#">DBGBCR&lt;n&gt;_EL1</a>	Debug Breakpoint Control Registers
10	000	0000	xxxx	110	<a href="#">DBGWVR&lt;n&gt;_EL1</a>	Debug Watchpoint Value Registers
10	000	0000	xxxx	111	<a href="#">DBGWCR&lt;n&gt;_EL1</a>	Debug Watchpoint Control Registers
10	000	0001	0000	000	<a href="#">MDRAR_EL1</a>	Monitor Debug ROM Address Register
10	000	0001	0000	100	<a href="#">OSLAR_EL1</a>	OS Lock Access Register
10	000	0001	0001	100	<a href="#">OSLSR_EL1</a>	OS Lock Status Register
10	000	0001	0011	100	<a href="#">OSDLR_EL1</a>	OS Double Lock Register
10	000	0001	0100	100	<a href="#">DBGPRCR_EL1</a>	Debug Power Control Register
10	000	0111	1000	110	<a href="#">DBGCLAIMSET_EL1</a>	Debug Claim Tag Set register
10	000	0111	1001	110	<a href="#">DBGCLAIMCLR_EL1</a>	Debug Claim Tag Clear register
10	000	0111	1110	110	<a href="#">DBGAUTHSTATUS_EL1</a>	Debug Authentication Status register
11	000	0000	0000	000	<a href="#">MIDR_EL1</a>	Main ID Register

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
11	001	0000	0000	000	<a href="#">CCSIDR_EL1</a>	Current Cache Size ID Register
11	010	0000	0000	000	<a href="#">CSSELR_EL1</a>	Cache Size Selection Register
11	100	0000	0000	000	<a href="#">VPIDR_EL2</a>	Virtualization Processor ID Register
11	001	0000	0000	001	<a href="#">CLIDR_EL1</a>	Cache Level ID Register
11	011	0000	0000	001	<a href="#">CTR_EL0</a>	Cache Type Register
11	000	0000	0000	101	<a href="#">MPIDR_EL1</a>	Multiprocessor Affinity Register
11	100	0000	0000	101	<a href="#">VMPIDR_EL2</a>	Virtualization Multiprocessor ID Register
11	000	0000	0000	110	<a href="#">REVIDR_EL1</a>	Revision ID Register
11	001	0000	0000	111	<a href="#">AIDR_EL1</a>	Auxiliary ID Register
11	011	0000	0000	111	<a href="#">DCZID_EL0</a>	Data Cache Zero ID register
11	000	0000	0001	000	<a href="#">ID_PFR0_EL1</a>	AArch32 Processor Feature Register 0
11	000	0000	0001	001	<a href="#">ID_PFR1_EL1</a>	AArch32 Processor Feature Register 1
11	000	0000	0001	010	<a href="#">ID_DFR0_EL1</a>	AArch32 Debug Feature Register 0
11	000	0000	0001	011	<a href="#">ID_AFR0_EL1</a>	AArch32 Auxiliary Feature Register 0
11	000	0000	0001	100	<a href="#">ID_MMFR0_EL1</a>	AArch32 Memory Model Feature Register 0
11	000	0000	0001	101	<a href="#">ID_MMFR1_EL1</a>	AArch32 Memory Model Feature Register 1
11	000	0000	0001	110	<a href="#">ID_MMFR2_EL1</a>	AArch32 Memory Model Feature Register 2
11	000	0000	0001	111	<a href="#">ID_MMFR3_EL1</a>	AArch32 Memory Model Feature Register 3
11	000	0000	0010	000	<a href="#">ID_ISAR0_EL1</a>	AArch32 Instruction Set Attribute Register 0
11	000	0000	0010	001	<a href="#">ID_ISAR1_EL1</a>	AArch32 Instruction Set Attribute Register 1
11	000	0000	0010	010	<a href="#">ID_ISAR2_EL1</a>	AArch32 Instruction Set Attribute Register 2
11	000	0000	0010	011	<a href="#">ID_ISAR3_EL1</a>	AArch32 Instruction Set Attribute Register 3
11	000	0000	0010	100	<a href="#">ID_ISAR4_EL1</a>	AArch32 Instruction Set Attribute Register 4
11	000	0000	0010	101	<a href="#">ID_ISAR5_EL1</a>	AArch32 Instruction Set Attribute Register 5
11	000	0000	0010	110	<a href="#">ID_MMFR4_EL1</a>	AArch32 Memory Model Feature Register 4
11	000	0000	0011	000	<a href="#">MVFR0_EL1</a>	AArch32 Media and VFP Feature Register 0
11	000	0000	0011	001	<a href="#">MVFR1_EL1</a>	AArch32 Media and VFP Feature Register 1
11	000	0000	0011	010	<a href="#">MVFR2_EL1</a>	AArch32 Media and VFP Feature Register 2
11	000	0000	0100	000	<a href="#">ID_AA64PFR0_EL1</a>	AArch64 Processor Feature Register 0
11	000	0000	0100	001	<a href="#">ID_AA64PFR1_EL1</a>	AArch64 Processor Feature Register 1
11	000	0000	0101	000	<a href="#">ID_AA64DFR0_EL1</a>	AArch64 Debug Feature Register 0
11	000	0000	0101	001	<a href="#">ID_AA64DFR1_EL1</a>	AArch64 Debug Feature Register 1
11	000	0000	0101	100	<a href="#">ID_AA64AFR0_EL1</a>	AArch64 Auxiliary Feature Register 0
11	000	0000	0101	101	<a href="#">ID_AA64AFR1_EL1</a>	AArch64 Auxiliary Feature Register 1
11	000	0000	0110	000	<a href="#">ID_AA64ISAR0_EL1</a>	AArch64 Instruction Set Attribute Register 0
11	000	0000	0110	001	<a href="#">ID_AA64ISAR1_EL1</a>	AArch64 Instruction Set Attribute Register 1
11	000	0000	0111	000	<a href="#">ID_AA64MMFR0_EL1</a>	AArch64 Memory Model Feature Register 0



op0	Register selectors		CRm	op2	Name	Description
	op1	CRn				
11	000	0000	0111	001	<a href="#">ID_AA64MMFR1_EL1</a>	AArch64 Memory Model Feature Register 1
11	000	0000	0111	010	<a href="#">ID_AA64MMFR2_EL1</a>	AArch64 Memory Model Feature Register 2
11	000	0001	0000	000	<a href="#">SCTLR_EL1</a>	System Control Register (EL1)
11	100	0001	0000	000	<a href="#">SCTLR_EL2</a>	System Control Register (EL2)
11	110	0001	0000	000	<a href="#">SCTLR_EL3</a>	System Control Register (EL3)
11	000	0001	0000	001	<a href="#">ACTLR_EL1</a>	Auxiliary Control Register (EL1)
11	100	0001	0000	001	<a href="#">ACTLR_EL2</a>	Auxiliary Control Register (EL2)
11	110	0001	0000	001	<a href="#">ACTLR_EL3</a>	Auxiliary Control Register (EL3)
11	000	0001	0000	010	<a href="#">CPACR_EL1</a>	Architectural Feature Access Control Register
11	100	0001	0001	000	<a href="#">HCR_EL2</a>	Hypervisor Configuration Register
11	110	0001	0001	000	<a href="#">SCR_EL3</a>	Secure Configuration Register
11	100	0001	0001	001	<a href="#">MDCR_EL2</a>	Monitor Debug Configuration Register (EL2)
11	110	0001	0001	001	<a href="#">SDER32_EL3</a>	AArch32 Secure Debug Enable Register
11	100	0001	0001	010	<a href="#">CPTR_EL2</a>	Architectural Feature Trap Register (EL2)
11	110	0001	0001	010	<a href="#">CPTR_EL3</a>	Architectural Feature Trap Register (EL3)
11	100	0001	0001	011	<a href="#">HSTR_EL2</a>	Hypervisor System Trap Register
11	100	0001	0001	111	<a href="#">HACR_EL2</a>	Hypervisor Auxiliary Control Register
11	110	0001	0011	001	<a href="#">MDCR_EL3</a>	Monitor Debug Configuration Register (EL3)
11	000	0010	0000	000	<a href="#">TTBR0_EL1</a>	Translation Table Base Register 0 (EL1)
11	100	0010	0000	000	<a href="#">TTBR0_EL2</a>	Translation Table Base Register 0 (EL2)
11	110	0010	0000	000	<a href="#">TTBR0_EL3</a>	Translation Table Base Register 0 (EL3)
11	000	0010	0000	001	<a href="#">TTBR1_EL1</a>	Translation Table Base Register 1 (EL1)
11	100	0010	0000	001	<a href="#">TTBR1_EL2</a>	Translation Table Base Register 1 (EL2)
11	000	0010	0000	010	<a href="#">TCR_EL1</a>	Translation Control Register (EL1)
11	100	0010	0000	010	<a href="#">TCR_EL2</a>	Translation Control Register (EL2)
11	110	0010	0000	010	<a href="#">TCR_EL3</a>	Translation Control Register (EL3)
11	100	0010	0001	000	<a href="#">VTTBR_EL2</a>	Virtualization Translation Table Base Register
11	100	0010	0001	010	<a href="#">VTCR_EL2</a>	Virtualization Translation Control Register
11	100	0011	0000	000	<a href="#">DACR32_EL2</a>	Domain Access Control Register
11	000	0100	0000	000	<a href="#">SPSR_EL1</a>	Saved Program Status Register (EL1)
11	100	0100	0000	000	<a href="#">SPSR_EL2</a>	Saved Program Status Register (EL2)
11	110	0100	0000	000	<a href="#">SPSR_EL3</a>	Saved Program Status Register (EL3)
11	000	0100	0000	001	<a href="#">ELR_EL1</a>	Exception Link Register (EL1)
11	100	0100	0000	001	<a href="#">ELR_EL2</a>	Exception Link Register (EL2)
11	110	0100	0000	001	<a href="#">ELR_EL3</a>	Exception Link Register (EL3)
11	000	0100	0001	000	<a href="#">SP_EL0</a>	Stack Pointer (EL0)
11	100	0100	0001	000	<a href="#">SP_EL1</a>	Stack Pointer (EL1)
11	110	0100	0001	000	<a href="#">SP_EL2</a>	Stack Pointer (EL2)
11	000	0100	0010	000	<a href="#">SPSel</a>	Stack Pointer Select
11	011	0100	0010	000	<a href="#">NZCV</a>	Condition Flags
11	011	0100	0010	001	<a href="#">DAIF</a>	Interrupt Mask Bits
11	000	0100	0010	010	<a href="#">CurrentEL</a>	Current Exception Level
11	000	0100	0010	011	<a href="#">PAN</a>	Privileged Access Never
11	000	0100	0010	100	<a href="#">UAO</a>	User Access Override
11	100	0100	0011	000	<a href="#">SPSR_irq</a>	Saved Program Status Register (IRQ mode)

op0	Register selectors		CRm	op2	Name	Description
	op1	CRn				
11	100	0100	0011	001	<a href="#">SPSR_abt</a>	Saved Program Status Register (Abort mode)
11	100	0100	0011	010	<a href="#">SPSR_und</a>	Saved Program Status Register (Undefined mode)
11	100	0100	0011	011	<a href="#">SPSR_fiq</a>	Saved Program Status Register (FIQ mode)
11	011	0100	0100	000	<a href="#">FPCR</a>	Floating-point Control Register
11	011	0100	0100	001	<a href="#">FPSR</a>	Floating-point Status Register
11	011	0100	0101	000	<a href="#">DPSR_EL0</a>	Debug Saved Program Status Register
11	011	0100	0101	001	<a href="#">DLR_EL0</a>	Debug Link Register
11	000	0100	0110	000	<a href="#">ICC_PMR_EL1</a>	Interrupt Controller Interrupt Priority Mask Register
11	000	0100	0110	000	<a href="#">ICV_PMR_EL1</a>	Interrupt Controller Virtual Interrupt Priority Mask Register
11	100	0101	0000	001	<a href="#">IFSR32_EL2</a>	Instruction Fault Status Register (EL2)
11	000	0101	0001	000	<a href="#">AFSR0_EL1</a>	Auxiliary Fault Status Register 0 (EL1)
11	100	0101	0001	000	<a href="#">AFSR0_EL2</a>	Auxiliary Fault Status Register 0 (EL2)
11	110	0101	0001	000	<a href="#">AFSR0_EL3</a>	Auxiliary Fault Status Register 0 (EL3)
11	000	0101	0001	001	<a href="#">AFSR1_EL1</a>	Auxiliary Fault Status Register 1 (EL1)
11	100	0101	0001	001	<a href="#">AFSR1_EL2</a>	Auxiliary Fault Status Register 1 (EL2)
11	110	0101	0001	001	<a href="#">AFSR1_EL3</a>	Auxiliary Fault Status Register 1 (EL3)
11	000	0101	0010	000	<a href="#">ESR_EL1</a>	Exception Syndrome Register (EL1)
11	100	0101	0010	000	<a href="#">ESR_EL2</a>	Exception Syndrome Register (EL2)
11	110	0101	0010	000	<a href="#">ESR_EL3</a>	Exception Syndrome Register (EL3)
11	100	0101	0011	000	<a href="#">FPEXC32_EL2</a>	Floating-Point Exception Control register
11	000	0110	0000	000	<a href="#">FAR_EL1</a>	Fault Address Register (EL1)
11	100	0110	0000	000	<a href="#">FAR_EL2</a>	Fault Address Register (EL2)
11	110	0110	0000	000	<a href="#">FAR_EL3</a>	Fault Address Register (EL3)
11	100	0110	0000	100	<a href="#">HPFAR_EL2</a>	Hypervisor IPA Fault Address Register
11	000	0111	0100	000	<a href="#">PAR_EL1</a>	Physical Address Register
11	011	1001	1100	000	<a href="#">PMCR_EL0</a>	Performance Monitors Control Register
11	011	1001	1100	001	<a href="#">PMCNTENSET_EL0</a>	Performance Monitors Count Enable Set register
11	011	1001	1100	010	<a href="#">PMCNTENCLR_EL0</a>	Performance Monitors Count Enable Clear register
11	011	1001	1100	011	<a href="#">PMOVSCCLR_EL0</a>	Performance Monitors Overflow Flag Status Clear Register
11	011	1001	1100	100	<a href="#">PMSWINC_EL0</a>	Performance Monitors Software Increment register
11	011	1001	1100	101	<a href="#">PMSELR_EL0</a>	Performance Monitors Event Counter Selection Register
11	011	1001	1100	110	<a href="#">PMCEID0_EL0</a>	Performance Monitors Common Event Identification register 0
11	011	1001	1100	111	<a href="#">PMCEID1_EL0</a>	Performance Monitors Common Event Identification register 1
11	011	1001	1101	000	<a href="#">PMCCNTR_EL0</a>	Performance Monitors Cycle Count Register
11	011	1001	1101	001	<a href="#">PMXEVTYPER_EL0</a>	Performance Monitors Selected Event Type Register
11	011	1001	1101	010	<a href="#">PMXVCNTR_EL0</a>	Performance Monitors Selected Event Count Register
11	011	1001	1110	000	<a href="#">PMUSERENR_EL0</a>	Performance Monitors User Enable Register
11	000	1001	1110	001	<a href="#">PMINTENSET_EL1</a>	Performance Monitors Interrupt Enable Set register
11	000	1001	1110	010	<a href="#">PMINTENCLR_EL1</a>	Performance Monitors Interrupt Enable Clear register

op0	Register selectors		CRm	op2	Name	Description
	op1	CRn				
11	011	1001	1110	011	<a href="#">PMOVSSET_EL0</a>	Performance Monitors Overflow Flag Status Set register
11	000	1010	0010	000	<a href="#">MAIR_EL1</a>	Memory Attribute Indirection Register (EL1)
11	100	1010	0010	000	<a href="#">MAIR_EL2</a>	Memory Attribute Indirection Register (EL2)
11	110	1010	0010	000	<a href="#">MAIR_EL3</a>	Memory Attribute Indirection Register (EL3)
11	000	1010	0011	000	<a href="#">AMAIR_EL1</a>	Auxiliary Memory Attribute Indirection Register (EL1)
11	100	1010	0011	000	<a href="#">AMAIR_EL2</a>	Auxiliary Memory Attribute Indirection Register (EL2)
11	110	1010	0011	000	<a href="#">AMAIR_EL3</a>	Auxiliary Memory Attribute Indirection Register (EL3)
11	000	1010	0100	000	<a href="#">LORSA_EL1</a>	LORegion Start Address (EL1)
11	000	1010	0100	001	<a href="#">LOREA_EL1</a>	LORegion End Address (EL1)
11	000	1010	0100	010	<a href="#">LORN_EL1</a>	LORegion Number (EL1)
11	000	1010	0100	011	<a href="#">LORC_EL1</a>	LORegion Control (EL1)
11	000	1010	0100	111	<a href="#">LORID_EL1</a>	LORegionID (EL1)
11	000	1100	0000	000	<a href="#">VBAR_EL1</a>	Vector Base Address Register (EL1)
11	100	1100	0000	000	<a href="#">VBAR_EL2</a>	Vector Base Address Register (EL2)
11	110	1100	0000	000	<a href="#">VBAR_EL3</a>	Vector Base Address Register (EL3)
11	000	1100	0000	001	<a href="#">RVBAR_EL1</a>	Reset Vector Base Address Register (if EL2 and EL3 not implemented)
11	100	1100	0000	001	<a href="#">RVBAR_EL2</a>	Reset Vector Base Address Register (if EL3 not implemented)
11	110	1100	0000	001	<a href="#">RVBAR_EL3</a>	Reset Vector Base Address Register (if EL3 implemented)
11	000	1100	0000	010	<a href="#">RMR_EL1</a>	Reset Management Register (EL1)
11	100	1100	0000	010	<a href="#">RMR_EL2</a>	Reset Management Register (EL2)
11	110	1100	0000	010	<a href="#">RMR_EL3</a>	Reset Management Register (EL3)
11	000	1100	0001	000	<a href="#">ISR_EL1</a>	Interrupt Status Register
11	000	1100	1000	000	<a href="#">ICC_IAR0_EL1</a>	Interrupt Controller Interrupt Acknowledge Register 0
11	000	1100	1000	000	<a href="#">ICV_IAR0_EL1</a>	Interrupt Controller Virtual Interrupt Acknowledge Register 0
11	000	1100	1000	001	<a href="#">ICC_EOIR0_EL1</a>	Interrupt Controller End Of Interrupt Register 0
11	000	1100	1000	001	<a href="#">ICV_EOIR0_EL1</a>	Interrupt Controller Virtual End Of Interrupt Register 0
11	000	1100	1000	010	<a href="#">ICC_HPPIR0_EL1</a>	Interrupt Controller Highest Priority Pending Interrupt Register 0
11	000	1100	1000	010	<a href="#">ICV_HPPIR0_EL1</a>	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
11	000	1100	1000	011	<a href="#">ICC_BPR0_EL1</a>	Interrupt Controller Binary Point Register 0
11	000	1100	1000	011	<a href="#">ICV_BPR0_EL1</a>	Interrupt Controller Virtual Binary Point Register 0
11	100	1100	1000	0xx	<a href="#">ICH_AP0R&lt;n&gt;_EL2</a>	Interrupt Controller Hyp Active Priorities Group 0 Registers
11	000	1100	1000	1xx	<a href="#">ICC_AP0R&lt;n&gt;_EL1</a>	Interrupt Controller Active Priorities Group 0 Registers
11	000	1100	1000	1xx	<a href="#">ICV_AP0R&lt;n&gt;_EL1</a>	Interrupt Controller Virtual Active Priorities Group 0 Registers
11	000	1100	1001	0xx	<a href="#">ICC_APIR&lt;n&gt;_EL1</a>	Interrupt Controller Active Priorities Group 1 Registers
11	000	1100	1001	0xx	<a href="#">ICV_APIR&lt;n&gt;_EL1</a>	Interrupt Controller Virtual Active Priorities Group 1 Registers

op0	Register selectors		CRm	op2	Name	Description
	op1	CRn				
11	100	1100	1001	0xx	<a href="#">ICH_APIR&lt;n&gt;_EL2</a>	Interrupt Controller Hyp Active Priorities Group 1 Registers
11	100	1100	1001	101	<a href="#">ICC_SRE_EL2</a>	Interrupt Controller System Register Enable register (EL2)
11	100	1100	1011	000	<a href="#">ICH_HCR_EL2</a>	Interrupt Controller Hyp Control Register
11	000	1100	1011	001	<a href="#">ICC_DIR_EL1</a>	Interrupt Controller Deactivate Interrupt Register
11	000	1100	1011	001	<a href="#">ICV_DIR_EL1</a>	Interrupt Controller Deactivate Virtual Interrupt Register
11	100	1100	1011	001	<a href="#">ICH_VTR_EL2</a>	Interrupt Controller VGIC Type Register
11	100	1100	1011	010	<a href="#">ICH_MISR_EL2</a>	Interrupt Controller Maintenance Interrupt State Register
11	000	1100	1011	011	<a href="#">ICC_RPR_EL1</a>	Interrupt Controller Running Priority Register
11	000	1100	1011	011	<a href="#">ICV_RPR_EL1</a>	Interrupt Controller Virtual Running Priority Register
11	100	1100	1011	011	<a href="#">ICH_EISR_EL2</a>	Interrupt Controller End of Interrupt Status Register
11	000	1100	1011	101	<a href="#">ICC_SGI1R_EL1</a>	Interrupt Controller Software Generated Interrupt Group 1 Register
11	100	1100	1011	101	<a href="#">ICH_ELRSR_EL2</a>	Interrupt Controller Empty List Register Status Register
11	000	1100	1011	110	<a href="#">ICC_ASGI1R_EL1</a>	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
11	000	1100	1011	111	<a href="#">ICC_SGI0R_EL1</a>	Interrupt Controller Software Generated Interrupt Group 0 Register
11	100	1100	1011	111	<a href="#">ICH_VMCR_EL2</a>	Interrupt Controller Virtual Machine Control Register
11	000	1100	1100	000	<a href="#">ICC_IAR1_EL1</a>	Interrupt Controller Interrupt Acknowledge Register 1
11	000	1100	1100	000	<a href="#">ICV_IAR1_EL1</a>	Interrupt Controller Virtual Interrupt Acknowledge Register 1
11	000	1100	1100	001	<a href="#">ICC_EOIR1_EL1</a>	Interrupt Controller End Of Interrupt Register 1
11	000	1100	1100	001	<a href="#">ICV_EOIR1_EL1</a>	Interrupt Controller Virtual End Of Interrupt Register 1
11	000	1100	1100	010	<a href="#">ICC_HPIR1_EL1</a>	Interrupt Controller Highest Priority Pending Interrupt Register 1
11	000	1100	1100	010	<a href="#">ICV_HPIR1_EL1</a>	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
11	000	1100	1100	011	<a href="#">ICC_BPR1_EL1</a>	Interrupt Controller Binary Point Register 1
11	000	1100	1100	011	<a href="#">ICV_BPR1_EL1</a>	Interrupt Controller Virtual Binary Point Register 1
11	000	1100	1100	100	<a href="#">ICC_CTLR_EL1</a>	Interrupt Controller Control Register (EL1)
11	000	1100	1100	100	<a href="#">ICV_CTLR_EL1</a>	Interrupt Controller Virtual Control Register
11	110	1100	1100	100	<a href="#">ICC_CTLR_EL3</a>	Interrupt Controller Control Register (EL3)
11	000	1100	1100	101	<a href="#">ICC_SRE_EL1</a>	Interrupt Controller System Register Enable register (EL1)
11	110	1100	1100	101	<a href="#">ICC_SRE_EL3</a>	Interrupt Controller System Register Enable register (EL3)
11	000	1100	1100	110	<a href="#">ICC_IGRPEN0_EL1</a>	Interrupt Controller Interrupt Group 0 Enable register
11	000	1100	1100	110	<a href="#">ICV_IGRPEN0_EL1</a>	Interrupt Controller Virtual Interrupt Group 0 Enable register
11	000	1100	1100	111	<a href="#">ICC_IGRPEN1_EL1</a>	Interrupt Controller Interrupt Group 1 Enable register
11	000	1100	1100	111	<a href="#">ICV_IGRPEN1_EL1</a>	Interrupt Controller Virtual Interrupt Group 1 Enable register

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
11	110	1100	1100	111	<a href="#">ICC_IGRPEN1_EL3</a>	Interrupt Controller Interrupt Group 1 Enable register (EL3)
11	100	1100	110x	xxx	<a href="#">ICH_LR&lt;n&gt;_EL2</a>	Interrupt Controller List Registers
11	000	1101	0000	001	<a href="#">CONTEXTIDR_EL1</a>	Context ID Register (EL1)
11	100	1101	0000	001	<a href="#">CONTEXTIDR_EL2</a>	Context ID Register (EL2)
11	011	1101	0000	010	<a href="#">TPIDR_EL0</a>	EL0 Read/Write Software Thread ID Register
11	100	1101	0000	010	<a href="#">TPIDR_EL2</a>	EL2 Software Thread ID Register
11	110	1101	0000	010	<a href="#">TPIDR_EL3</a>	EL3 Software Thread ID Register
11	011	1101	0000	011	<a href="#">TPIDRRO_EL0</a>	EL0 Read-Only Software Thread ID Register
11	000	1101	0000	100	<a href="#">TPIDR_EL1</a>	EL1 Software Thread ID Register
11	011	1110	0000	000	<a href="#">CNTFRQ_EL0</a>	Counter-timer Frequency register
11	011	1110	0000	001	<a href="#">CNTPCT_EL0</a>	Counter-timer Physical Count register
11	011	1110	0000	010	<a href="#">CNTVCT_EL0</a>	Counter-timer Virtual Count register
11	100	1110	0000	011	<a href="#">CNTVOFF_EL2</a>	Counter-timer Virtual Offset register
11	000	1110	0001	000	<a href="#">CNTKCTL_EL1</a>	Counter-timer Kernel Control register
11	100	1110	0001	000	<a href="#">CNTHCTL_EL2</a>	Counter-timer Hypervisor Control register
11	011	1110	0010	000	<a href="#">CNTP_TVAL_EL0</a>	Counter-timer Physical Timer TimerValue register
11	100	1110	0010	000	<a href="#">CNTHP_TVAL_EL2</a>	Counter-timer Hypervisor Physical Timer TimerValue register
11	111	1110	0010	000	<a href="#">CNTPS_TVAL_EL1</a>	Counter-timer Physical Secure Timer TimerValue register
11	011	1110	0010	001	<a href="#">CNTP_CTL_EL0</a>	Counter-timer Physical Timer Control register
11	100	1110	0010	001	<a href="#">CNTHP_CTL_EL2</a>	Counter-timer Hypervisor Physical Timer Control register
11	111	1110	0010	001	<a href="#">CNTPS_CTL_EL1</a>	Counter-timer Physical Secure Timer Control register
11	011	1110	0010	010	<a href="#">CNTP_CVAL_EL0</a>	Counter-timer Physical Timer CompareValue register
11	100	1110	0010	010	<a href="#">CNTHP_CVAL_EL2</a>	Counter-timer Hypervisor Physical Timer CompareValue register
11	111	1110	0010	010	<a href="#">CNTPS_CVAL_EL1</a>	Counter-timer Physical Secure Timer CompareValue register
11	011	1110	0011	000	<a href="#">CNTV_TVAL_EL0</a>	Counter-timer Virtual Timer TimerValue register
11	100	1110	0011	000	<a href="#">CNTHV_TVAL_EL2</a>	Counter-timer Virtual Timer TimerValue register (EL2)
11	011	1110	0011	001	<a href="#">CNTV_CTL_EL0</a>	Counter-timer Virtual Timer Control register
11	100	1110	0011	001	<a href="#">CNTHV_CTL_EL2</a>	Counter-timer Virtual Timer Control register (EL2)
11	011	1110	0011	010	<a href="#">CNTV_CVAL_EL0</a>	Counter-timer Virtual Timer CompareValue register
11	100	1110	0011	010	<a href="#">CNTHV_CVAL_EL2</a>	Counter-timer Virtual Timer CompareValue register (EL2)
11	011	1110	10xx	xxx	<a href="#">PMEVCNTR&lt;n&gt;_EL0</a>	Performance Monitors Event Count Registers
11	011	1110	1111	111	<a href="#">PMCCFILTR_EL0</a>	Performance Monitors Cycle Count Filter Register
11	011	1110	11xx	xxx	<a href="#">PMEVTYPER&lt;n&gt;_EL0</a>	Performance Monitors Event Type Registers
11	xxx	1x11	xxxx	xxx	<a href="#">S3_&lt;op1&gt;_&lt;Cn&gt;_&lt;Cm&gt;_&lt;op2&gt;</a>	IMPLEMENTATION DEFINED registers

**Accessed using TLBI:**

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
01	100	1000	0000	001	<a href="#">TLBI IPAS2E1IS</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
01	100	1000	0000	101	<a href="#">TLBI IPAS2LE1IS</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
01	000	1000	0011	000	<a href="#">TLBI VMALLE1IS</a>	TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable
01	100	1000	0011	000	<a href="#">TLBI ALLE2IS</a>	TLB Invalidate All, EL2, Inner Shareable
01	110	1000	0011	000	<a href="#">TLBI ALLE3IS</a>	TLB Invalidate All, EL3, Inner Shareable
01	000	1000	0011	001	<a href="#">TLBI VAE1IS</a>	TLB Invalidate by VA, EL1, Inner Shareable
01	100	1000	0011	001	<a href="#">TLBI VAE2IS</a>	TLB Invalidate by VA, EL2, Inner Shareable
01	110	1000	0011	001	<a href="#">TLBI VAE3IS</a>	TLB Invalidate by VA, EL3, Inner Shareable
01	000	1000	0011	010	<a href="#">TLBI ASIDE1IS</a>	TLB Invalidate by ASID, EL1, Inner Shareable
01	000	1000	0011	011	<a href="#">TLBI VAAE1IS</a>	TLB Invalidate by VA, All ASID, EL1, Inner Shareable
01	100	1000	0011	100	<a href="#">TLBI ALLE1IS</a>	TLB Invalidate All, EL1, Inner Shareable
01	000	1000	0011	101	<a href="#">TLBI VALE1IS</a>	TLB Invalidate by VA, Last level, EL1, Inner Shareable
01	100	1000	0011	101	<a href="#">TLBI VALE2IS</a>	TLB Invalidate by VA, Last level, EL2, Inner Shareable
01	110	1000	0011	101	<a href="#">TLBI VALE3IS</a>	TLB Invalidate by VA, Last level, EL3, Inner Shareable
01	100	1000	0011	110	<a href="#">TLBI VMALLS12E1IS</a>	TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner Shareable
01	000	1000	0011	111	<a href="#">TLBI VAALE1IS</a>	TLB Invalidate by VA, All ASID, EL1, Last Level, Inner Shareable
01	100	1000	0100	001	<a href="#">TLBI IPAS2E1</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1
01	100	1000	0100	101	<a href="#">TLBI IPAS2LE1</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
01	000	1000	0111	000	<a href="#">TLBI VMALLE1</a>	TLB Invalidate by VMID, All at stage 1, EL1
01	100	1000	0111	000	<a href="#">TLBI ALLE2</a>	TLB Invalidate All, EL2
01	110	1000	0111	000	<a href="#">TLBI ALLE3</a>	TLB Invalidate All, EL3
01	000	1000	0111	001	<a href="#">TLBI VAE1</a>	TLB Invalidate by VA, EL1
01	100	1000	0111	001	<a href="#">TLBI VAE2</a>	TLB Invalidate by VA, EL2
01	110	1000	0111	001	<a href="#">TLBI VAE3</a>	TLB Invalidate by VA, EL3
01	000	1000	0111	010	<a href="#">TLBI ASIDE1</a>	TLB Invalidate by ASID, EL1
01	000	1000	0111	011	<a href="#">TLBI VAAE1</a>	TLB Invalidate by VA, All ASID, EL1
01	100	1000	0111	100	<a href="#">TLBI ALLE1</a>	TLB Invalidate All, EL1
01	000	1000	0111	101	<a href="#">TLBI VALE1</a>	TLB Invalidate by VA, Last level, EL1
01	100	1000	0111	101	<a href="#">TLBI VALE2</a>	TLB Invalidate by VA, Last level, EL2
01	110	1000	0111	101	<a href="#">TLBI VALE3</a>	TLB Invalidate by VA, Last level, EL3
01	100	1000	0111	110	<a href="#">TLBI VMALLS12E1</a>	TLB Invalidate by VMID, All at Stage 1 and 2, EL1
01	000	1000	0111	111	<a href="#">TLBI VAALE1</a>	TLB Invalidate by VA, All ASID, Last level, EL1

**Accessed using SYSL/SYS:**

Register selectors				Name	Description
op1	CRn	CRm	op2		
xxx	1x11	xxxx	xxx	<a href="#">S1_&lt;op1&gt;_&lt;Cn&gt;_&lt;Cm&gt;_&lt;op2&gt;</a>	IMPLEMENTATION DEFINED maintenance instructions

**Accessed using DC/IC:**

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
01	000	0111	0001	000	<a href="#">IC IALLUIS</a>	Instruction Cache Invalidate All to PoU, Inner Shareable
01	011	0111	0100	001	<a href="#">DC ZVA</a>	Data Cache Zero by VA
01	000	0111	0101	000	<a href="#">IC IALLU</a>	Instruction Cache Invalidate All to PoU
01	011	0111	0101	001	<a href="#">IC IVAU</a>	Instruction Cache line Invalidate by VA to PoU
01	000	0111	0110	001	<a href="#">DC IVAC</a>	Data or unified Cache line Invalidate by VA to PoC
01	000	0111	0110	010	<a href="#">DC ISW</a>	Data or unified Cache line Invalidate by Set/Way
01	011	0111	1010	001	<a href="#">DC CVAC</a>	Data or unified Cache line Clean by VA to PoC
01	000	0111	1010	010	<a href="#">DC CSW</a>	Data or unified Cache line Clean by Set/Way
01	011	0111	1011	001	<a href="#">DC CVAU</a>	Data or unified Cache line Clean by VA to PoU
01	011	0111	1100	001	<a href="#">DC CVAP</a>	Data or unified Cache line Clean by VA to PoP
01	011	0111	1110	001	<a href="#">DC CIVAC</a>	Data or unified Cache line Clean and Invalidate by VA to PoC
01	000	0111	1110	010	<a href="#">DC CISW</a>	Data or unified Cache line Clean and Invalidate by Set/Way

**Accessed using AT:**

Register selectors					Name	Description
op0	op1	CRn	CRm	op2		
01	000	0111	1000	000	<a href="#">AT S1E1R</a>	Address Translate Stage 1 EL1 Read
01	100	0111	1000	000	<a href="#">AT S1E2R</a>	Address Translate Stage 1 EL2 Read
01	110	0111	1000	000	<a href="#">AT S1E3R</a>	Address Translate Stage 1 EL3 Read
01	000	0111	1000	001	<a href="#">AT S1E1W</a>	Address Translate Stage 1 EL1 Write
01	100	0111	1000	001	<a href="#">AT S1E2W</a>	Address Translate Stage 1 EL2 Write
01	110	0111	1000	001	<a href="#">AT S1E3W</a>	Address Translate Stage 1 EL3 Write
01	000	0111	1000	010	<a href="#">AT S1E0R</a>	Address Translate Stage 1 EL0 Read
01	000	0111	1000	011	<a href="#">AT S1E0W</a>	Address Translate Stage 1 EL0 Write
01	100	0111	1000	100	<a href="#">AT S12E1R</a>	Address Translate Stages 1 and 2 EL1 Read
01	100	0111	1000	101	<a href="#">AT S12E1W</a>	Address Translate Stages 1 and 2 EL1 Write
01	100	0111	1000	110	<a href="#">AT S12E0R</a>	Address Translate Stages 1 and 2 EL0 Read
01	100	0111	1000	111	<a href="#">AT S12E0W</a>	Address Translate Stages 1 and 2 EL0 Write
01	000	0111	1001	000	<a href="#">AT S1E1RP</a>	Address Translate Stage 1 EL1 Read PAN
01	000	0111	1001	001	<a href="#">AT S1E1WP</a>	Address Translate Stage 1 EL1 Write PAN

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# 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](#)

## In the ID functional group:

Exec state	Name	Description
AArch32	<a href="#">AIDR</a>	Auxiliary ID Register
AArch32	<a href="#">CCSIDR</a>	Current Cache Size ID Register
AArch32	<a href="#">CLIDR</a>	Cache Level ID Register
AArch32	<a href="#">CSSELR</a>	Cache Size Selection Register
AArch32	<a href="#">CTR</a>	Cache Type Register
AArch32	<a href="#">FPSID</a>	Floating-Point System ID register
AArch32	<a href="#">ID_AFR0</a>	Auxiliary Feature Register 0
AArch32	<a href="#">ID_DFR0</a>	Debug Feature Register 0
AArch32	<a href="#">ID_ISAR0</a>	Instruction Set Attribute Register 0
AArch32	<a href="#">ID_ISAR1</a>	Instruction Set Attribute Register 1
AArch32	<a href="#">ID_ISAR2</a>	Instruction Set Attribute Register 2
AArch32	<a href="#">ID_ISAR3</a>	Instruction Set Attribute Register 3
AArch32	<a href="#">ID_ISAR4</a>	Instruction Set Attribute Register 4
AArch32	<a href="#">ID_ISAR5</a>	Instruction Set Attribute Register 5
AArch32	<a href="#">ID_MMFR0</a>	Memory Model Feature Register 0
AArch32	<a href="#">ID_MMFR1</a>	Memory Model Feature Register 1
AArch32	<a href="#">ID_MMFR2</a>	Memory Model Feature Register 2
AArch32	<a href="#">ID_MMFR3</a>	Memory Model Feature Register 3
AArch32	<a href="#">ID_MMFR4</a>	Memory Model Feature Register 4
AArch32	<a href="#">ID_PFR0</a>	Processor Feature Register 0
AArch32	<a href="#">ID_PFR1</a>	Processor Feature Register 1
AArch32	<a href="#">MIDR</a>	Main ID Register
AArch32	<a href="#">MPIDR</a>	Multiprocessor Affinity Register
AArch32	<a href="#">MVFR0</a>	Media and VFP Feature Register 0
AArch32	<a href="#">MVFR1</a>	Media and VFP Feature Register 1
AArch32	<a href="#">MVFR2</a>	Media and VFP Feature Register 2
AArch32	<a href="#">REVIDR</a>	Revision ID Register
AArch32	<a href="#">TCMTR</a>	TCM Type Register



Exec state	Name	Description
AArch32	<a href="#">TLBTR</a>	TLB Type Register
AArch32	<a href="#">VMPIDR</a>	Virtualization Multiprocessor ID Register
AArch32	<a href="#">VPIDR</a>	Virtualization Processor ID Register
AArch64	<a href="#">AIDR_EL1</a>	Auxiliary ID Register
AArch64	<a href="#">CCSIDR_EL1</a>	Current Cache Size ID Register
AArch64	<a href="#">CLIDR_EL1</a>	Cache Level ID Register
AArch64	<a href="#">CSSELR_EL1</a>	Cache Size Selection Register
AArch64	<a href="#">CTR_EL0</a>	Cache Type Register
AArch64	<a href="#">DCZID_EL0</a>	Data Cache Zero ID register
AArch64	<a href="#">ID_AA64AFR0_EL1</a>	AArch64 Auxiliary Feature Register 0
AArch64	<a href="#">ID_AA64AFR1_EL1</a>	AArch64 Auxiliary Feature Register 1
AArch64	<a href="#">ID_AA64DFR0_EL1</a>	AArch64 Debug Feature Register 0
AArch64	<a href="#">ID_AA64DFR1_EL1</a>	AArch64 Debug Feature Register 1
AArch64	<a href="#">ID_AA64ISAR0_EL1</a>	AArch64 Instruction Set Attribute Register 0
AArch64	<a href="#">ID_AA64ISAR1_EL1</a>	AArch64 Instruction Set Attribute Register 1
AArch64	<a href="#">ID_AA64MMFR0_EL1</a>	AArch64 Memory Model Feature Register 0
AArch64	<a href="#">ID_AA64MMFR1_EL1</a>	AArch64 Memory Model Feature Register 1
AArch64	<a href="#">ID_AA64MMFR2_EL1</a>	AArch64 Memory Model Feature Register 2
AArch64	<a href="#">ID_AA64PFR0_EL1</a>	AArch64 Processor Feature Register 0
AArch64	<a href="#">ID_AA64PFR1_EL1</a>	AArch64 Processor Feature Register 1
AArch64	<a href="#">ID_AFR0_EL1</a>	AArch32 Auxiliary Feature Register 0
AArch64	<a href="#">ID_DFR0_EL1</a>	AArch32 Debug Feature Register 0
AArch64	<a href="#">ID_ISAR0_EL1</a>	AArch32 Instruction Set Attribute Register 0
AArch64	<a href="#">ID_ISAR1_EL1</a>	AArch32 Instruction Set Attribute Register 1
AArch64	<a href="#">ID_ISAR2_EL1</a>	AArch32 Instruction Set Attribute Register 2
AArch64	<a href="#">ID_ISAR3_EL1</a>	AArch32 Instruction Set Attribute Register 3
AArch64	<a href="#">ID_ISAR4_EL1</a>	AArch32 Instruction Set Attribute Register 4
AArch64	<a href="#">ID_ISAR5_EL1</a>	AArch32 Instruction Set Attribute Register 5
AArch64	<a href="#">ID_MMFR0_EL1</a>	AArch32 Memory Model Feature Register 0
AArch64	<a href="#">ID_MMFR1_EL1</a>	AArch32 Memory Model Feature Register 1
AArch64	<a href="#">ID_MMFR2_EL1</a>	AArch32 Memory Model Feature Register 2
AArch64	<a href="#">ID_MMFR3_EL1</a>	AArch32 Memory Model Feature Register 3
AArch64	<a href="#">ID_MMFR4_EL1</a>	AArch32 Memory Model Feature Register 4
AArch64	<a href="#">ID_PFR0_EL1</a>	AArch32 Processor Feature Register 0
AArch64	<a href="#">ID_PFR1_EL1</a>	AArch32 Processor Feature Register 1
AArch64	<a href="#">MIDR_EL1</a>	Main ID Register
AArch64	<a href="#">MPIDR_EL1</a>	Multiprocessor Affinity Register
AArch64	<a href="#">MVFR0_EL1</a>	AArch32 Media and VFP Feature Register 0
AArch64	<a href="#">MVFR1_EL1</a>	AArch32 Media and VFP Feature Register 1
AArch64	<a href="#">MVFR2_EL1</a>	AArch32 Media and VFP Feature Register 2
AArch64	<a href="#">REVIDR_EL1</a>	Revision ID Register
AArch64	<a href="#">VMPIDR_EL2</a>	Virtualization Multiprocessor ID Register
AArch64	<a href="#">VPIDR_EL2</a>	Virtualization Processor ID Register
External	<a href="#">EDAA32PFR</a>	External Debug AArch32 Processor Feature Register
External	<a href="#">EDDFR</a>	External Debug Feature Register
External	<a href="#">EDPFR</a>	External Debug Processor Feature Register
External	<a href="#">MIDR_EL1</a>	Main ID Register

## In the Memory functional group:

Exec state	Name	Description
AArch32	<a href="#">AMAIRO</a>	Auxiliary Memory Attribute Indirection Register 0
AArch32	<a href="#">AMAIR1</a>	Auxiliary Memory Attribute Indirection Register 1
AArch32	<a href="#">CONTEXTIDR</a>	Context ID Register
AArch32	<a href="#">DACR</a>	Domain Access Control Register
AArch32	<a href="#">HAMAIRO</a>	Hyp Auxiliary Memory Attribute Indirection Register 0
AArch32	<a href="#">HAMAIR1</a>	Hyp Auxiliary Memory Attribute Indirection Register 1
AArch32	<a href="#">HMAIRO</a>	Hyp Memory Attribute Indirection Register 0
AArch32	<a href="#">HMAIR1</a>	Hyp Memory Attribute Indirection Register 1
AArch32	<a href="#">HTCR</a>	Hyp Translation Control Register
AArch32	<a href="#">HTTBR</a>	Hyp Translation Table Base Register
AArch32	<a href="#">MAIRO</a>	Memory Attribute Indirection Register 0
AArch32	<a href="#">MAIR1</a>	Memory Attribute Indirection Register 1

Exec state	Name	Description
AArch32	<a href="#">NMRR</a>	Normal Memory Remap Register
AArch32	<a href="#">PRRR</a>	Primary Region Remap Register
AArch32	<a href="#">TTBCR</a>	Translation Table Base Control Register
AArch32	<a href="#">TTBCR2</a>	Translation Table Base Control Register 2
AArch32	<a href="#">TTBR0</a>	Translation Table Base Register 0
AArch32	<a href="#">TTBR1</a>	Translation Table Base Register 1
AArch32	<a href="#">VTCT</a>	Virtualization Translation Control Register
AArch32	<a href="#">VTTBR</a>	Virtualization Translation Table Base Register
AArch64	<a href="#">AMAIR_EL1</a>	Auxiliary Memory Attribute Indirection Register (EL1)
AArch64	<a href="#">AMAIR_EL2</a>	Auxiliary Memory Attribute Indirection Register (EL2)
AArch64	<a href="#">AMAIR_EL3</a>	Auxiliary Memory Attribute Indirection Register (EL3)
AArch64	<a href="#">CONTEXTIDR_EL1</a>	Context ID Register (EL1)
AArch64	<a href="#">CONTEXTIDR_EL2</a>	Context ID Register (EL2)
AArch64	<a href="#">DACR32_EL2</a>	Domain Access Control Register
AArch64	<a href="#">LORC_EL1</a>	LORegion Control (EL1)
AArch64	<a href="#">LOREA_EL1</a>	LORegion End Address (EL1)
AArch64	<a href="#">LORID_EL1</a>	LORegionID (EL1)
AArch64	<a href="#">LORN_EL1</a>	LORegion Number (EL1)
AArch64	<a href="#">LORSA_EL1</a>	LORegion Start Address (EL1)
AArch64	<a href="#">MAIR_EL1</a>	Memory Attribute Indirection Register (EL1)
AArch64	<a href="#">MAIR_EL2</a>	Memory Attribute Indirection Register (EL2)
AArch64	<a href="#">MAIR_EL3</a>	Memory Attribute Indirection Register (EL3)
AArch64	<a href="#">TCR_EL1</a>	Translation Control Register (EL1)
AArch64	<a href="#">TCR_EL2</a>	Translation Control Register (EL2)
AArch64	<a href="#">TCR_EL3</a>	Translation Control Register (EL3)
AArch64	<a href="#">TTBR0_EL1</a>	Translation Table Base Register 0 (EL1)
AArch64	<a href="#">TTBR0_EL2</a>	Translation Table Base Register 0 (EL2)
AArch64	<a href="#">TTBR0_EL3</a>	Translation Table Base Register 0 (EL3)
AArch64	<a href="#">TTBR1_EL1</a>	Translation Table Base Register 1 (EL1)
AArch64	<a href="#">TTBR1_EL2</a>	Translation Table Base Register 1 (EL2)
AArch64	<a href="#">VTCT_EL2</a>	Virtualization Translation Control Register
AArch64	<a href="#">VTTBR_EL2</a>	Virtualization Translation Table Base Register

## In the Other functional group:

Exec state	Name	Description
AArch32	<a href="#">ACTLR</a>	Auxiliary Control Register
AArch32	<a href="#">ACTLR2</a>	Auxiliary Control Register 2
AArch32	<a href="#">CPACR</a>	Architectural Feature Access Control Register
AArch32	<a href="#">HACTLR</a>	Hyp Auxiliary Control Register
AArch32	<a href="#">HACTLR2</a>	Hyp Auxiliary Control Register 2
AArch32	<a href="#">HSCTLR</a>	Hyp System Control Register
AArch32	<a href="#">SCTLR</a>	System Control Register
AArch64	<a href="#">ACTLR_EL1</a>	Auxiliary Control Register (EL1)
AArch64	<a href="#">ACTLR_EL2</a>	Auxiliary Control Register (EL2)
AArch64	<a href="#">ACTLR_EL3</a>	Auxiliary Control Register (EL3)
AArch64	<a href="#">CPACR_EL1</a>	Architectural Feature Access Control Register
AArch64	<a href="#">SCTLR_EL1</a>	System Control Register (EL1)
AArch64	<a href="#">SCTLR_EL2</a>	System Control Register (EL2)
AArch64	<a href="#">SCTLR_EL3</a>	System Control Register (EL3)

## In the Exception functional group:

Exec state	Name	Description
AArch32	<a href="#">ADFSR</a>	Auxiliary Data Fault Status Register
AArch32	<a href="#">AIFSR</a>	Auxiliary Instruction Fault Status Register
AArch32	<a href="#">DFAR</a>	Data Fault Address Register
AArch32	<a href="#">DFSR</a>	Data Fault Status Register
AArch32	<a href="#">HADFSR</a>	Hyp Auxiliary Data Fault Status Register
AArch32	<a href="#">HAIFSR</a>	Hyp Auxiliary Instruction Fault Status Register
AArch32	<a href="#">HDFAR</a>	Hyp Data Fault Address Register
AArch32	<a href="#">HIFAR</a>	Hyp Instruction Fault Address Register

Exec state	Name	Description
AArch32	<a href="#">HPFAR</a>	Hyp IPA Fault Address Register
AArch32	<a href="#">HSR</a>	Hyp Syndrome Register
AArch32	<a href="#">HVBAR</a>	Hyp Vector Base Address Register
AArch32	<a href="#">IFAR</a>	Instruction Fault Address Register
AArch32	<a href="#">IFSR</a>	Instruction Fault Status Register
AArch32	<a href="#">ISR</a>	Interrupt Status Register
AArch32	<a href="#">MVBAR</a>	Monitor Vector Base Address Register
AArch32	<a href="#">VBAR</a>	Vector Base Address Register
AArch64	<a href="#">AFSR0_EL1</a>	Auxiliary Fault Status Register 0 (EL1)
AArch64	<a href="#">AFSR0_EL2</a>	Auxiliary Fault Status Register 0 (EL2)
AArch64	<a href="#">AFSR0_EL3</a>	Auxiliary Fault Status Register 0 (EL3)
AArch64	<a href="#">AFSR1_EL1</a>	Auxiliary Fault Status Register 1 (EL1)
AArch64	<a href="#">AFSR1_EL2</a>	Auxiliary Fault Status Register 1 (EL2)
AArch64	<a href="#">AFSR1_EL3</a>	Auxiliary Fault Status Register 1 (EL3)
AArch64	<a href="#">ESR_EL1</a>	Exception Syndrome Register (EL1)
AArch64	<a href="#">ESR_EL2</a>	Exception Syndrome Register (EL2)
AArch64	<a href="#">ESR_EL3</a>	Exception Syndrome Register (EL3)
AArch64	<a href="#">ESR_ELx</a>	Exception Syndrome Register (ELx)
AArch64	<a href="#">FAR_EL1</a>	Fault Address Register (EL1)
AArch64	<a href="#">FAR_EL2</a>	Fault Address Register (EL2)
AArch64	<a href="#">FAR_EL3</a>	Fault Address Register (EL3)
AArch64	<a href="#">HPFAR_EL2</a>	Hypervisor IPA Fault Address Register
AArch64	<a href="#">IFSR32_EL2</a>	Instruction Fault Status Register (EL2)
AArch64	<a href="#">ISR_EL1</a>	Interrupt Status Register
AArch64	<a href="#">VBAR_EL1</a>	Vector Base Address Register (EL1)
AArch64	<a href="#">VBAR_EL2</a>	Vector Base Address Register (EL2)
AArch64	<a href="#">VBAR_EL3</a>	Vector Base Address Register (EL3)

## In the Special functional group:

Exec state	Name	Description
AArch32	<a href="#">DLR</a>	Debug Link Register
AArch32	<a href="#">DSPSR</a>	Debug Saved Program Status Register
AArch32	<a href="#">ELR_hyp</a>	Exception Link Register (Hyp mode)
AArch32	<a href="#">FPSCR</a>	Floating-Point Status and Control Register
AArch32	<a href="#">SPSR</a>	Saved Program Status Register
AArch32	<a href="#">SPSR_abt</a>	Saved Program Status Register (Abort mode)
AArch32	<a href="#">SPSR_fiq</a>	Saved Program Status Register (FIQ mode)
AArch32	<a href="#">SPSR_hyp</a>	Saved Program Status Register (Hyp mode)
AArch32	<a href="#">SPSR_irq</a>	Saved Program Status Register (IRQ mode)
AArch32	<a href="#">SPSR_mon</a>	Saved Program Status Register (Monitor mode)
AArch32	<a href="#">SPSR_svc</a>	Saved Program Status Register (Supervisor mode)
AArch32	<a href="#">SPSR_und</a>	Saved Program Status Register (Undefined mode)
AArch64	<a href="#">DLR_EL0</a>	Debug Link Register
AArch64	<a href="#">DSPSR_EL0</a>	Debug Saved Program Status Register
AArch64	<a href="#">ELR_EL1</a>	Exception Link Register (EL1)
AArch64	<a href="#">ELR_EL2</a>	Exception Link Register (EL2)
AArch64	<a href="#">ELR_EL3</a>	Exception Link Register (EL3)
AArch64	<a href="#">FPCR</a>	Floating-point Control Register
AArch64	<a href="#">FPSR</a>	Floating-point Status Register
AArch64	<a href="#">SPSR_EL1</a>	Saved Program Status Register (EL1)
AArch64	<a href="#">SPSR_EL2</a>	Saved Program Status Register (EL2)
AArch64	<a href="#">SPSR_EL3</a>	Saved Program Status Register (EL3)
AArch64	<a href="#">SPSR_abt</a>	Saved Program Status Register (Abort mode)
AArch64	<a href="#">SPSR_fiq</a>	Saved Program Status Register (FIQ mode)
AArch64	<a href="#">SPSR_irq</a>	Saved Program Status Register (IRQ mode)
AArch64	<a href="#">SPSR_und</a>	Saved Program Status Register (Undefined mode)
AArch64	<a href="#">SP_EL0</a>	Stack Pointer (EL0)
AArch64	<a href="#">SP_EL1</a>	Stack Pointer (EL1)
AArch64	<a href="#">SP_EL2</a>	Stack Pointer (EL2)
AArch64	<a href="#">SP_EL3</a>	Stack Pointer (EL3)
AArch64	<a href="#">UAO</a>	User Access Override

## In the PSTATE functional group:

Exec state	Name	Description
AArch32	<a href="#">APSR</a>	Application Program Status Register
AArch32	<a href="#">CPSR</a>	Current Program Status Register
AArch64	<a href="#">CurrentEL</a>	Current Exception Level
AArch64	<a href="#">DAIF</a>	Interrupt Mask Bits
AArch64	<a href="#">NZCV</a>	Condition Flags
AArch64	<a href="#">PAN</a>	Privileged Access Never
AArch64	<a href="#">SPSel</a>	Stack Pointer Select

## In the Cache functional group:

Exec state	Name	Description
AArch32	<a href="#">BPIALL</a>	Branch Predictor Invalidate All
AArch32	<a href="#">BPIALLIS</a>	Branch Predictor Invalidate All, Inner Shareable
AArch32	<a href="#">BPIMVA</a>	Branch Predictor Invalidate by VA
AArch32	<a href="#">DCCIMVAC</a>	Data Cache line Clean and Invalidate by VA to PoC
AArch32	<a href="#">DCCISW</a>	Data Cache line Clean and Invalidate by Set/Way
AArch32	<a href="#">DCCMVAC</a>	Data Cache line Clean by VA to PoC
AArch32	<a href="#">DCCMVAU</a>	Data Cache line Clean by VA to PoU
AArch32	<a href="#">DCCSW</a>	Data Cache line Clean by Set/Way
AArch32	<a href="#">DCIMVAC</a>	Data Cache line Invalidate by VA to PoC
AArch32	<a href="#">DCISW</a>	Data Cache line Invalidate by Set/Way
AArch32	<a href="#">ICIALLU</a>	Instruction Cache Invalidate All to PoU
AArch32	<a href="#">ICIALLUIS</a>	Instruction Cache Invalidate All to PoU, Inner Shareable
AArch32	<a href="#">ICIMVAU</a>	Instruction Cache line Invalidate by VA to PoU
AArch64	<a href="#">DC CISW</a>	Data or unified Cache line Clean and Invalidate by Set/Way
AArch64	<a href="#">DC CIVAC</a>	Data or unified Cache line Clean and Invalidate by VA to PoC
AArch64	<a href="#">DC CSW</a>	Data or unified Cache line Clean by Set/Way
AArch64	<a href="#">DC CVAC</a>	Data or unified Cache line Clean by VA to PoC
AArch64	<a href="#">DC CVAP</a>	Data or unified Cache line Clean by VA to PoP
AArch64	<a href="#">DC CVAU</a>	Data or unified Cache line Clean by VA to PoU
AArch64	<a href="#">IC IALLU</a>	Instruction Cache Invalidate All to PoU
AArch64	<a href="#">IC IALLUIS</a>	Instruction Cache Invalidate All to PoU, Inner Shareable
AArch64	<a href="#">DC ISW</a>	Data or unified Cache line Invalidate by Set/Way
AArch64	<a href="#">DC IVAC</a>	Data or unified Cache line Invalidate by VA to PoC
AArch64	<a href="#">IC IVAU</a>	Instruction Cache line Invalidate by VA to PoU
AArch64	<a href="#">DC ZVA</a>	Data Cache Zero by VA

## In the Address functional group:

Exec state	Name	Description
AArch32	<a href="#">ATS12NSOPR</a>	Address Translate Stages 1 and 2 Non-secure Only PL1 Read
AArch32	<a href="#">ATS12NSOPW</a>	Address Translate Stages 1 and 2 Non-secure Only PL1 Write
AArch32	<a href="#">ATS12NSOUR</a>	Address Translate Stages 1 and 2 Non-secure Only Unprivileged Read
AArch32	<a href="#">ATS12NSOUW</a>	Address Translate Stages 1 and 2 Non-secure Only Unprivileged Write
AArch32	<a href="#">ATS1CPR</a>	Address Translate Stage 1 Current state PL1 Read
AArch32	<a href="#">ATS1CPRP</a>	Address Translate Stage 1 Current state PL1 Read PAN
AArch32	<a href="#">ATS1CPW</a>	Address Translate Stage 1 Current state PL1 Write
AArch32	<a href="#">ATS1CPWP</a>	Address Translate Stage 1 Current state PL1 Write PAN
AArch32	<a href="#">ATS1CUR</a>	Address Translate Stage 1 Current state Unprivileged Read
AArch32	<a href="#">ATS1CUW</a>	Address Translate Stage 1 Current state Unprivileged Write
AArch32	<a href="#">ATS1HR</a>	Address Translate Stage 1 Hyp mode Read
AArch32	<a href="#">ATS1HW</a>	Address Translate Stage 1 Hyp mode Write
AArch32	<a href="#">PAR</a>	Physical Address Register
AArch64	<a href="#">PAR_EL1</a>	Physical Address Register
AArch64	<a href="#">AT S12E0R</a>	Address Translate Stages 1 and 2 EL0 Read
AArch64	<a href="#">AT S12E0W</a>	Address Translate Stages 1 and 2 EL0 Write
AArch64	<a href="#">AT S12E1R</a>	Address Translate Stages 1 and 2 EL1 Read
AArch64	<a href="#">AT S12E1W</a>	Address Translate Stages 1 and 2 EL1 Write
AArch64	<a href="#">AT S1E0R</a>	Address Translate Stage 1 EL0 Read
AArch64	<a href="#">AT S1E0W</a>	Address Translate Stage 1 EL0 Write

Exec state	Name	Description
AArch64	<a href="#">AT S1E1R</a>	Address Translate Stage 1 EL1 Read
AArch64	<a href="#">AT S1E1RP</a>	Address Translate Stage 1 EL1 Read PAN
AArch64	<a href="#">AT S1E1W</a>	Address Translate Stage 1 EL1 Write
AArch64	<a href="#">AT S1E1WP</a>	Address Translate Stage 1 EL1 Write PAN
AArch64	<a href="#">AT S1E2R</a>	Address Translate Stage 1 EL2 Read
AArch64	<a href="#">AT S1E2W</a>	Address Translate Stage 1 EL2 Write
AArch64	<a href="#">AT S1E3R</a>	Address Translate Stage 1 EL3 Read
AArch64	<a href="#">AT S1E3W</a>	Address Translate Stage 1 EL3 Write

## In the TLB functional group:

Exec state	Name	Description
AArch32	<a href="#">DTLBIALL</a>	Data TLB Invalidate All
AArch32	<a href="#">DTLBIASID</a>	Data TLB Invalidate by ASID match
AArch32	<a href="#">DTLBIMVA</a>	Data TLB Invalidate by VA
AArch32	<a href="#">ITLBIALL</a>	Instruction TLB Invalidate All
AArch32	<a href="#">ITLBIASID</a>	Instruction TLB Invalidate by ASID match
AArch32	<a href="#">ITLBIMVA</a>	Instruction TLB Invalidate by VA
AArch32	<a href="#">TLBIALL</a>	TLB Invalidate All
AArch32	<a href="#">TLBIALLH</a>	TLB Invalidate All, Hyp mode
AArch32	<a href="#">TLBIALLHIS</a>	TLB Invalidate All, Hyp mode, Inner Shareable
AArch32	<a href="#">TLBIALLIS</a>	TLB Invalidate All, Inner Shareable
AArch32	<a href="#">TLBIALLNSNH</a>	TLB Invalidate All, Non-Secure Non-Hyp
AArch32	<a href="#">TLBIALLNSNHIS</a>	TLB Invalidate All, Non-Secure Non-Hyp, Inner Shareable
AArch32	<a href="#">TLBIASID</a>	TLB Invalidate by ASID match
AArch32	<a href="#">TLBIASIDIS</a>	TLB Invalidate by ASID match, Inner Shareable
AArch32	<a href="#">TLBIIPAS2</a>	TLB Invalidate by Intermediate Physical Address, Stage 2
AArch32	<a href="#">TLBIIPAS2IS</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Inner Shareable
AArch32	<a href="#">TLBIIPAS2L</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level
AArch32	<a href="#">TLBIIPAS2LIS</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, Inner Shareable
AArch32	<a href="#">TLBIMVA</a>	TLB Invalidate by VA
AArch32	<a href="#">TLBIMVAA</a>	TLB Invalidate by VA, All ASID
AArch32	<a href="#">TLBIMVAAIS</a>	TLB Invalidate by VA, All ASID, Inner Shareable
AArch32	<a href="#">TLBIMVAAL</a>	TLB Invalidate by VA, All ASID, Last level
AArch32	<a href="#">TLBIMVAALIS</a>	TLB Invalidate by VA, All ASID, Last level, Inner Shareable
AArch32	<a href="#">TLBIMVAH</a>	TLB Invalidate by VA, Hyp mode
AArch32	<a href="#">TLBIMVAHIS</a>	TLB Invalidate by VA, Hyp mode, Inner Shareable
AArch32	<a href="#">TLBIMVAIS</a>	TLB Invalidate by VA, Inner Shareable
AArch32	<a href="#">TLBIMVAL</a>	TLB Invalidate by VA, Last level
AArch32	<a href="#">TLBIMVALH</a>	TLB Invalidate by VA, Last level, Hyp mode
AArch32	<a href="#">TLBIMVALHIS</a>	TLB Invalidate by VA, Last level, Hyp mode, Inner Shareable
AArch32	<a href="#">TLBIMVALIS</a>	TLB Invalidate by VA, Last level, Inner Shareable
AArch64	<a href="#">TLBI ALLE1</a>	TLB Invalidate All, EL1
AArch64	<a href="#">TLBI ALLE1IS</a>	TLB Invalidate All, EL1, Inner Shareable
AArch64	<a href="#">TLBI ALLE2</a>	TLB Invalidate All, EL2
AArch64	<a href="#">TLBI ALLE2IS</a>	TLB Invalidate All, EL2, Inner Shareable
AArch64	<a href="#">TLBI ALLE3</a>	TLB Invalidate All, EL3
AArch64	<a href="#">TLBI ALLE3IS</a>	TLB Invalidate All, EL3, Inner Shareable
AArch64	<a href="#">TLBI ASIDE1</a>	TLB Invalidate by ASID, EL1
AArch64	<a href="#">TLBI ASIDE1IS</a>	TLB Invalidate by ASID, EL1, Inner Shareable
AArch64	<a href="#">TLBI IPAS2E1</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1
AArch64	<a href="#">TLBI IPAS2E1IS</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
AArch64	<a href="#">TLBI IPAS2LE1</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
AArch64	<a href="#">TLBI IPAS2LE1IS</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
AArch64	<a href="#">TLBI VAAE1</a>	TLB Invalidate by VA, All ASID, EL1
AArch64	<a href="#">TLBI VAAE1IS</a>	TLB Invalidate by VA, All ASID, EL1, Inner Shareable
AArch64	<a href="#">TLBI VAALE1</a>	TLB Invalidate by VA, All ASID, Last level, EL1
AArch64	<a href="#">TLBI VAALE1IS</a>	TLB Invalidate by VA, All ASID, EL1, Last Level, Inner Shareable
AArch64	<a href="#">TLBI VAE1</a>	TLB Invalidate by VA, EL1
AArch64	<a href="#">TLBI VAE1IS</a>	TLB Invalidate by VA, EL1, Inner Shareable
AArch64	<a href="#">TLBI VAE2</a>	TLB Invalidate by VA, EL2
AArch64	<a href="#">TLBI VAE2IS</a>	TLB Invalidate by VA, EL2, Inner Shareable
AArch64	<a href="#">TLBI VAE3</a>	TLB Invalidate by VA, EL3



Exec state	Name	Description
AArch64	<a href="#">TLBI VAE3IS</a>	TLB Invalidate by VA, EL3, Inner Shareable
AArch64	<a href="#">TLBI VALE1</a>	TLB Invalidate by VA, Last level, EL1
AArch64	<a href="#">TLBI VALE1IS</a>	TLB Invalidate by VA, Last level, EL1, Inner Shareable
AArch64	<a href="#">TLBI VALE2</a>	TLB Invalidate by VA, Last level, EL2
AArch64	<a href="#">TLBI VALE2IS</a>	TLB Invalidate by VA, Last level, EL2, Inner Shareable
AArch64	<a href="#">TLBI VALE3</a>	TLB Invalidate by VA, Last level, EL3
AArch64	<a href="#">TLBI VALE3IS</a>	TLB Invalidate by VA, Last level, EL3, Inner Shareable
AArch64	<a href="#">TLBI VMALLE1</a>	TLB Invalidate by VMID, All at stage 1, EL1
AArch64	<a href="#">TLBI VMALLE1IS</a>	TLB Invalidate by VMID, All at stage 1, EL1, Inner Shareable
AArch64	<a href="#">TLBI VMALLS12E1</a>	TLB Invalidate by VMID, All at Stage 1 and 2, EL1
AArch64	<a href="#">TLBI VMALLS12E1IS</a>	TLB Invalidate by VMID, All at Stage 1 and 2, EL1, Inner Shareable

## In the PMU functional group:

Exec state	Name	Description
AArch32	<a href="#">PMCCFILTR</a>	Performance Monitors Cycle Count Filter Register
AArch32	<a href="#">PMCCNTR</a>	Performance Monitors Cycle Count Register
AArch32	<a href="#">PMCEID0</a>	Performance Monitors Common Event Identification register 0
AArch32	<a href="#">PMCEID1</a>	Performance Monitors Common Event Identification register 1
AArch32	<a href="#">PMCEID2</a>	Performance Monitors Common Event Identification register 2
AArch32	<a href="#">PMCEID3</a>	Performance Monitors Common Event Identification register 3
AArch32	<a href="#">PMCNTENCLR</a>	Performance Monitors Count Enable Clear register
AArch32	<a href="#">PMCNTENSET</a>	Performance Monitors Count Enable Set register
AArch32	<a href="#">PMCR</a>	Performance Monitors Control Register
AArch32	<a href="#">PMEVCNTR&lt;n&gt;</a>	Performance Monitors Event Count Registers
AArch32	<a href="#">PMEVTYPER&lt;n&gt;</a>	Performance Monitors Event Type Registers
AArch32	<a href="#">PMINTENCLR</a>	Performance Monitors Interrupt Enable Clear register
AArch32	<a href="#">PMINTENSET</a>	Performance Monitors Interrupt Enable Set register
AArch32	<a href="#">PMOVSr</a>	Performance Monitors Overflow Flag Status Register
AArch32	<a href="#">PMOVSSET</a>	Performance Monitors Overflow Flag Status Set register
AArch32	<a href="#">PMSELR</a>	Performance Monitors Event Counter Selection Register
AArch32	<a href="#">PMSWINC</a>	Performance Monitors Software Increment register
AArch32	<a href="#">PMUSERENR</a>	Performance Monitors User Enable Register
AArch32	<a href="#">PMXEVCNTR</a>	Performance Monitors Selected Event Count Register
AArch32	<a href="#">PMXEVTYPER</a>	Performance Monitors Selected Event Type Register
AArch64	<a href="#">PMCCFILTR_EL0</a>	Performance Monitors Cycle Count Filter Register
AArch64	<a href="#">PMCCNTR_EL0</a>	Performance Monitors Cycle Count Register
AArch64	<a href="#">PMCEID0_EL0</a>	Performance Monitors Common Event Identification register 0
AArch64	<a href="#">PMCEID1_EL0</a>	Performance Monitors Common Event Identification register 1
AArch64	<a href="#">PMCNTENCLR_EL0</a>	Performance Monitors Count Enable Clear register
AArch64	<a href="#">PMCNTENSET_EL0</a>	Performance Monitors Count Enable Set register
AArch64	<a href="#">PMCR_EL0</a>	Performance Monitors Control Register
AArch64	<a href="#">PMEVCNTR&lt;n&gt;_EL0</a>	Performance Monitors Event Count Registers
AArch64	<a href="#">PMEVTYPER&lt;n&gt;_EL0</a>	Performance Monitors Event Type Registers
AArch64	<a href="#">PMINTENCLR_EL1</a>	Performance Monitors Interrupt Enable Clear register
AArch64	<a href="#">PMINTENSET_EL1</a>	Performance Monitors Interrupt Enable Set register
AArch64	<a href="#">PMOVSLR_EL0</a>	Performance Monitors Overflow Flag Status Clear Register
AArch64	<a href="#">PMOVSSET_EL0</a>	Performance Monitors Overflow Flag Status Set register
AArch64	<a href="#">PMSELR_EL0</a>	Performance Monitors Event Counter Selection Register
AArch64	<a href="#">PMSWINC_EL0</a>	Performance Monitors Software Increment register
AArch64	<a href="#">PMUSERENR_EL0</a>	Performance Monitors User Enable Register
AArch64	<a href="#">PMXEVCNTR_EL0</a>	Performance Monitors Selected Event Count Register
AArch64	<a href="#">PMXEVTYPER_EL0</a>	Performance Monitors Selected Event Type Register
External	<a href="#">PMAUTHSTATUS</a>	Performance Monitors Authentication Status register
External	<a href="#">PMCCFILTR_EL0</a>	Performance Monitors Cycle Counter Filter Register
External	<a href="#">PMCCNTR_EL0</a>	Performance Monitors Cycle Counter
External	<a href="#">PMCEID0</a>	Performance Monitors Common Event Identification register 0
External	<a href="#">PMCEID1</a>	Performance Monitors Common Event Identification register 1
External	<a href="#">PMCEID2</a>	Performance Monitors Common Event Identification register 2
External	<a href="#">PMCEID3</a>	Performance Monitors Common Event Identification register 3
External	<a href="#">PMCFGR</a>	Performance Monitors Configuration Register
External	<a href="#">PMCID1SR</a>	CONTEXTIDR_EL1 Sample Register
External	<a href="#">PMCID2SR</a>	CONTEXTIDR_EL2 Sample Register

Exec state	Name	Description
External	<a href="#">PMCIDR0</a>	Performance Monitors Component Identification Register 0
External	<a href="#">PMCIDR1</a>	Performance Monitors Component Identification Register 1
External	<a href="#">PMCIDR2</a>	Performance Monitors Component Identification Register 2
External	<a href="#">PMCIDR3</a>	Performance Monitors Component Identification Register 3
External	<a href="#">PMCNTENCLR_EL0</a>	Performance Monitors Count Enable Clear register
External	<a href="#">PMCNTENSET_EL0</a>	Performance Monitors Count Enable Set register
External	<a href="#">PMCR_EL0</a>	Performance Monitors Control Register
External	<a href="#">PMDEVAFF0</a>	Performance Monitors Device Affinity register 0
External	<a href="#">PMDEVAFF1</a>	Performance Monitors Device Affinity register 1
External	<a href="#">PMDEVARCH</a>	Performance Monitors Device Architecture register
External	<a href="#">PMDEVID</a>	Performance Monitors Device ID register
External	<a href="#">PMDEVTYPE</a>	Performance Monitors Device Type register
External	<a href="#">PMEVCNTR&lt;n&gt;_EL0</a>	Performance Monitors Event Count Registers
External	<a href="#">PMEVTYPEPER&lt;n&gt;_EL0</a>	Performance Monitors Event Type Registers
External	<a href="#">PMINTENCLR_EL1</a>	Performance Monitors Interrupt Enable Clear register
External	<a href="#">PMINTENSET_EL1</a>	Performance Monitors Interrupt Enable Set register
External	<a href="#">PMITCTRL</a>	Performance Monitors Integration mode Control register
External	<a href="#">PMLAR</a>	Performance Monitors Lock Access Register
External	<a href="#">PMLSR</a>	Performance Monitors Lock Status Register
External	<a href="#">PMOVSCLR_EL0</a>	Performance Monitors Overflow Flag Status Clear register
External	<a href="#">PMOVSSET_EL0</a>	Performance Monitors Overflow Flag Status Set register
External	<a href="#">PMPCSR</a>	Program Counter Sample Register
External	<a href="#">PMPIDR0</a>	Performance Monitors Peripheral Identification Register 0
External	<a href="#">PMPIDR1</a>	Performance Monitors Peripheral Identification Register 1
External	<a href="#">PMPIDR2</a>	Performance Monitors Peripheral Identification Register 2
External	<a href="#">PMPIDR3</a>	Performance Monitors Peripheral Identification Register 3
External	<a href="#">PMPIDR4</a>	Performance Monitors Peripheral Identification Register 4
External	<a href="#">PMSWINC_EL0</a>	Performance Monitors Software Increment register
External	<a href="#">PMVIDSR</a>	VMID Sample Register

## In the Reset functional group:

Exec state	Name	Description
AArch32	<a href="#">HRMR</a>	Hyp Reset Management Register
AArch32	<a href="#">RMR</a>	Reset Management Register
AArch32	<a href="#">RVBAR</a>	Reset Vector Base Address Register
AArch64	<a href="#">RMR_EL1</a>	Reset Management Register (EL1)
AArch64	<a href="#">RMR_EL2</a>	Reset Management Register (EL2)
AArch64	<a href="#">RMR_EL3</a>	Reset Management Register (EL3)
AArch64	<a href="#">RVBAR_EL1</a>	Reset Vector Base Address Register (if EL2 and EL3 not implemented)
AArch64	<a href="#">RVBAR_EL2</a>	Reset Vector Base Address Register (if EL3 not implemented)
AArch64	<a href="#">RVBAR_EL3</a>	Reset Vector Base Address Register (if EL3 implemented)

## In the Thread functional group:

Exec state	Name	Description
AArch32	<a href="#">HTPIDR</a>	Hyp Software Thread ID Register
AArch32	<a href="#">TPIDRPRW</a>	PL1 Software Thread ID Register
AArch32	<a href="#">TPIDRURO</a>	PL0 Read-Only Software Thread ID Register
AArch32	<a href="#">TPIDRURW</a>	PL0 Read/Write Software Thread ID Register
AArch64	<a href="#">TPIDRRO_EL0</a>	EL0 Read-Only Software Thread ID Register
AArch64	<a href="#">TPIDR_EL0</a>	EL0 Read/Write Software Thread ID Register
AArch64	<a href="#">TPIDR_EL1</a>	EL1 Software Thread ID Register
AArch64	<a href="#">TPIDR_EL2</a>	EL2 Software Thread ID Register
AArch64	<a href="#">TPIDR_EL3</a>	EL3 Software Thread ID Register

## In the IMP DEF functional group:

Exec state	Name	Description
AArch32	<a href="#">ACTLR</a>	Auxiliary Control Register
AArch32	<a href="#">ACTLR2</a>	Auxiliary Control Register 2

Exec state	Name	Description
AArch32	<a href="#">ADFSR</a>	Auxiliary Data Fault Status Register
AArch32	<a href="#">AIDR</a>	Auxiliary ID Register
AArch32	<a href="#">AIFSR</a>	Auxiliary Instruction Fault Status Register
AArch32	<a href="#">AMAIRO</a>	Auxiliary Memory Attribute Indirection Register 0
AArch32	<a href="#">AMAIR1</a>	Auxiliary Memory Attribute Indirection Register 1
AArch32	<a href="#">HACTLR</a>	Hyp Auxiliary Control Register
AArch32	<a href="#">HACTLR2</a>	Hyp Auxiliary Control Register 2
AArch32	<a href="#">HADFSR</a>	Hyp Auxiliary Data Fault Status Register
AArch32	<a href="#">HAIFSR</a>	Hyp Auxiliary Instruction Fault Status Register
AArch32	<a href="#">HAMAIRO</a>	Hyp Auxiliary Memory Attribute Indirection Register 0
AArch32	<a href="#">HAMAIR1</a>	Hyp Auxiliary Memory Attribute Indirection Register 1
AArch64	<a href="#">ACTLR_EL1</a>	Auxiliary Control Register (EL1)
AArch64	<a href="#">ACTLR_EL2</a>	Auxiliary Control Register (EL2)
AArch64	<a href="#">ACTLR_EL3</a>	Auxiliary Control Register (EL3)
AArch64	<a href="#">AFSR0_EL1</a>	Auxiliary Fault Status Register 0 (EL1)
AArch64	<a href="#">AFSR0_EL2</a>	Auxiliary Fault Status Register 0 (EL2)
AArch64	<a href="#">AFSR0_EL3</a>	Auxiliary Fault Status Register 0 (EL3)
AArch64	<a href="#">AFSR1_EL1</a>	Auxiliary Fault Status Register 1 (EL1)
AArch64	<a href="#">AFSR1_EL2</a>	Auxiliary Fault Status Register 1 (EL2)
AArch64	<a href="#">AFSR1_EL3</a>	Auxiliary Fault Status Register 1 (EL3)
AArch64	<a href="#">AIDR_EL1</a>	Auxiliary ID Register
AArch64	<a href="#">AMAIR_EL1</a>	Auxiliary Memory Attribute Indirection Register (EL1)
AArch64	<a href="#">AMAIR_EL2</a>	Auxiliary Memory Attribute Indirection Register (EL2)
AArch64	<a href="#">AMAIR_EL3</a>	Auxiliary Memory Attribute Indirection Register (EL3)
AArch64	<a href="#">HACR_EL2</a>	Hypervisor Auxiliary Control Register
AArch64	<a href="#">S1_&lt;op1&gt;_&lt;Cn&gt;_&lt;Cm&gt;_&lt;op2&gt;</a>	IMPLEMENTATION DEFINED maintenance instructions
AArch64	<a href="#">S3_&lt;op1&gt;_&lt;Cn&gt;_&lt;Cm&gt;_&lt;op2&gt;</a>	IMPLEMENTATION DEFINED registers

## In the Timer functional group:

Exec state	Name	Description
AArch32	<a href="#">CNTFRQ</a>	Counter-timer Frequency register
AArch32	<a href="#">CNTHCTL</a>	Counter-timer Hyp Control register
AArch32	<a href="#">CNTHP_CTL</a>	Counter-timer Hyp Physical Timer Control register
AArch32	<a href="#">CNTHP_CVAL</a>	Counter-timer Hyp Physical Timer CompareValue register
AArch32	<a href="#">CNTHP_TVAL</a>	Counter-timer Hyp Physical Timer TimerValue register
AArch32	<a href="#">CNTHV_CTL</a>	Counter-timer Virtual Timer Control register (EL2)
AArch32	<a href="#">CNTHV_CVAL</a>	Counter-timer Virtual Timer CompareValue register (EL2)
AArch32	<a href="#">CNTHV_TVAL</a>	Counter-timer Virtual Timer TimerValue register (EL2)
AArch32	<a href="#">CNTKCTL</a>	Counter-timer Kernel Control register
AArch32	<a href="#">CNTPCT</a>	Counter-timer Physical Count register
AArch32	<a href="#">CNTP_CTL</a>	Counter-timer Physical Timer Control register
AArch32	<a href="#">CNTP_CVAL</a>	Counter-timer Physical Timer CompareValue register
AArch32	<a href="#">CNTP_TVAL</a>	Counter-timer Physical Timer TimerValue register
AArch32	<a href="#">CNTVCT</a>	Counter-timer Virtual Count register
AArch32	<a href="#">CNTVOFF</a>	Counter-timer Virtual Offset register
AArch32	<a href="#">CNTV_CTL</a>	Counter-timer Virtual Timer Control register
AArch32	<a href="#">CNTV_CVAL</a>	Counter-timer Virtual Timer CompareValue register
AArch32	<a href="#">CNTV_TVAL</a>	Counter-timer Virtual Timer TimerValue register
AArch64	<a href="#">CNTFRQ_EL0</a>	Counter-timer Frequency register
AArch64	<a href="#">CNTHCTL_EL2</a>	Counter-timer Hypervisor Control register
AArch64	<a href="#">CNTHP_CTL_EL2</a>	Counter-timer Hypervisor Physical Timer Control register
AArch64	<a href="#">CNTHP_CVAL_EL2</a>	Counter-timer Hypervisor Physical Timer CompareValue register
AArch64	<a href="#">CNTHP_TVAL_EL2</a>	Counter-timer Hypervisor Physical Timer TimerValue register
AArch64	<a href="#">CNTHV_CTL_EL2</a>	Counter-timer Virtual Timer Control register (EL2)
AArch64	<a href="#">CNTHV_CVAL_EL2</a>	Counter-timer Virtual Timer CompareValue register (EL2)
AArch64	<a href="#">CNTHV_TVAL_EL2</a>	Counter-timer Virtual Timer TimerValue register (EL2)
AArch64	<a href="#">CNTKCTL_EL1</a>	Counter-timer Kernel Control register
AArch64	<a href="#">CNTPCT_EL0</a>	Counter-timer Physical Count register
AArch64	<a href="#">CNTPS_CTL_EL1</a>	Counter-timer Physical Secure Timer Control register
AArch64	<a href="#">CNTPS_CVAL_EL1</a>	Counter-timer Physical Secure Timer CompareValue register
AArch64	<a href="#">CNTPS_TVAL_EL1</a>	Counter-timer Physical Secure Timer TimerValue register
AArch64	<a href="#">CNTP_CTL_EL0</a>	Counter-timer Physical Timer Control register



Exec state	Name	Description
AArch64	<a href="#">CNTP_CVAL_EL0</a>	Counter-timer Physical Timer CompareValue register
AArch64	<a href="#">CNTP_TVAL_EL0</a>	Counter-timer Physical Timer TimerValue register
AArch64	<a href="#">CNTVCT_EL0</a>	Counter-timer Virtual Count register
AArch64	<a href="#">CNTVOFF_EL2</a>	Counter-timer Virtual Offset register
AArch64	<a href="#">CNTV_CTL_EL0</a>	Counter-timer Virtual Timer Control register
AArch64	<a href="#">CNTV_CVAL_EL0</a>	Counter-timer Virtual Timer CompareValue register
AArch64	<a href="#">CNTV_TVAL_EL0</a>	Counter-timer Virtual Timer TimerValue register
External	<a href="#">CNTACR&lt;n&gt;</a>	Counter-timer Access Control Registers
External	<a href="#">CNTCR</a>	Counter Control Register
External	<a href="#">CNTCV</a>	Counter Count Value register
External	<a href="#">CNTEL0ACR</a>	Counter-timer EL0 Access Control Register
External	<a href="#">CNTFID0</a>	Counter Frequency ID
External	<a href="#">CNTFID&lt;n&gt;</a>	Counter Frequency IDs
External	<a href="#">CNTFRQ</a>	Counter-timer Frequency
External	<a href="#">CNTNSAR</a>	Counter-timer Non-secure Access Register
External	<a href="#">CNTPCT</a>	Counter-timer Physical Count
External	<a href="#">CNTP_CTL</a>	Counter-timer Physical Timer Control
External	<a href="#">CNTP_CVAL</a>	Counter-timer Physical Timer CompareValue
External	<a href="#">CNTP_TVAL</a>	Counter-timer Physical Timer TimerValue
External	<a href="#">CNTSR</a>	Counter Status Register
External	<a href="#">CNTTIDR</a>	Counter-timer Timer ID Register
External	<a href="#">CNTVCT</a>	Counter-timer Virtual Count
External	<a href="#">CNTVOFF</a>	Counter-timer Virtual Offset
External	<a href="#">CNTVOFF&lt;n&gt;</a>	Counter-timer Virtual Offsets
External	<a href="#">CNTV_CTL</a>	Counter-timer Virtual Timer Control
External	<a href="#">CNTV_CVAL</a>	Counter-timer Virtual Timer CompareValue
External	<a href="#">CNTV_TVAL</a>	Counter-timer Virtual Timer TimerValue
External	<a href="#">CounterID&lt;n&gt;</a>	Counter ID registers

## In the Debug functional group:

Exec state	Name	Description
AArch32	<a href="#">DBGAUTHSTATUS</a>	Debug Authentication Status register
AArch32	<a href="#">DBGBCR&lt;n&gt;</a>	Debug Breakpoint Control Registers
AArch32	<a href="#">DBGBVR&lt;n&gt;</a>	Debug Breakpoint Value Registers
AArch32	<a href="#">DBGBXVR&lt;n&gt;</a>	Debug Breakpoint Extended Value Registers
AArch32	<a href="#">DBGCLAIMCLR</a>	Debug Claim Tag Clear register
AArch32	<a href="#">DBGCLAIMSET</a>	Debug Claim Tag Set register
AArch32	<a href="#">DBGDCCINT</a>	DCC Interrupt Enable Register
AArch32	<a href="#">DBGDEVID</a>	Debug Device ID register 0
AArch32	<a href="#">DBGDEVID1</a>	Debug Device ID register 1
AArch32	<a href="#">DBGDEVID2</a>	Debug Device ID register 2
AArch32	<a href="#">DBGDIDR</a>	Debug ID Register
AArch32	<a href="#">DBGDRAR</a>	Debug ROM Address Register
AArch32	<a href="#">DBGDSAR</a>	Debug Self Address Register
AArch32	<a href="#">DBGDSCRext</a>	Debug Status and Control Register, External View
AArch32	<a href="#">DBGDSCRint</a>	Debug Status and Control Register, Internal View
AArch32	<a href="#">DBGDTRRXext</a>	Debug OS Lock Data Transfer Register, Receive, External View
AArch32	<a href="#">DBGDTRRXint</a>	Debug Data Transfer Register, Receive
AArch32	<a href="#">DBGDTRTXext</a>	Debug OS Lock Data Transfer Register, Transmit
AArch32	<a href="#">DBGDTRTXint</a>	Debug Data Transfer Register, Transmit
AArch32	<a href="#">DBGOSDLR</a>	Debug OS Double Lock Register
AArch32	<a href="#">DBGOSECCR</a>	Debug OS Lock Exception Catch Control Register
AArch32	<a href="#">DBGOSLAR</a>	Debug OS Lock Access Register
AArch32	<a href="#">DBGOSLSR</a>	Debug OS Lock Status Register
AArch32	<a href="#">DBGPRCR</a>	Debug Power Control Register
AArch32	<a href="#">DBGVCR</a>	Debug Vector Catch Register
AArch32	<a href="#">DBGWCR&lt;n&gt;</a>	Debug Watchpoint Control Registers
AArch32	<a href="#">DBGWFR</a>	Debug Watchpoint Fault Address Register
AArch32	<a href="#">DBGWVR&lt;n&gt;</a>	Debug Watchpoint Value Registers
AArch32	<a href="#">DLR</a>	Debug Link Register
AArch32	<a href="#">DSPSR</a>	Debug Saved Program Status Register
AArch32	<a href="#">HDCR</a>	Hyp Debug Control Register

Exec state	Name	Description
AArch32	<a href="#">SDCR</a>	Secure Debug Control Register
AArch32	<a href="#">SDER</a>	Secure Debug Enable Register
AArch64	<a href="#">DBGAUTHSTATUS_EL1</a>	Debug Authentication Status register
AArch64	<a href="#">DBGBCR&lt;n&gt;_EL1</a>	Debug Breakpoint Control Registers
AArch64	<a href="#">DBGBVR&lt;n&gt;_EL1</a>	Debug Breakpoint Value Registers
AArch64	<a href="#">DBGCLAIMCLR_EL1</a>	Debug Claim Tag Clear register
AArch64	<a href="#">DBGCLAIMSET_EL1</a>	Debug Claim Tag Set register
AArch64	<a href="#">DBGDTRRX_EL0</a>	Debug Data Transfer Register, Receive
AArch64	<a href="#">DBGDTRTX_EL0</a>	Debug Data Transfer Register, Transmit
AArch64	<a href="#">DBGDTR_EL0</a>	Debug Data Transfer Register, half-duplex
AArch64	<a href="#">DBGPRCR_EL1</a>	Debug Power Control Register
AArch64	<a href="#">DBGVCR32_EL2</a>	Debug Vector Catch Register
AArch64	<a href="#">DBGWCR&lt;n&gt;_EL1</a>	Debug Watchpoint Control Registers
AArch64	<a href="#">DBGWVR&lt;n&gt;_EL1</a>	Debug Watchpoint Value Registers
AArch64	<a href="#">DLR_EL0</a>	Debug Link Register
AArch64	<a href="#">DSPSR_EL0</a>	Debug Saved Program Status Register
AArch64	<a href="#">MDCCINT_EL1</a>	Monitor DCC Interrupt Enable Register
AArch64	<a href="#">MDCCSR_EL0</a>	Monitor DCC Status Register
AArch64	<a href="#">MDCR_EL2</a>	Monitor Debug Configuration Register (EL2)
AArch64	<a href="#">MDCR_EL3</a>	Monitor Debug Configuration Register (EL3)
AArch64	<a href="#">MDRAR_EL1</a>	Monitor Debug ROM Address Register
AArch64	<a href="#">MDSCR_EL1</a>	Monitor Debug System Control Register
AArch64	<a href="#">OSDLR_EL1</a>	OS Double Lock Register
AArch64	<a href="#">OSDTRRX_EL1</a>	OS Lock Data Transfer Register, Receive
AArch64	<a href="#">OSDTRTX_EL1</a>	OS Lock Data Transfer Register, Transmit
AArch64	<a href="#">OSECCR_EL1</a>	OS Lock Exception Catch Control Register
AArch64	<a href="#">OSLAR_EL1</a>	OS Lock Access Register
AArch64	<a href="#">OSLSR_EL1</a>	OS Lock Status Register
AArch64	<a href="#">SDER32_EL3</a>	AArch32 Secure Debug Enable Register
External	<a href="#">DBGAUTHSTATUS_EL1</a>	Debug Authentication Status register
External	<a href="#">DBGBCR&lt;n&gt;_EL1</a>	Debug Breakpoint Control Registers
External	<a href="#">DBGBVR&lt;n&gt;_EL1</a>	Debug Breakpoint Value Registers
External	<a href="#">DBGCLAIMCLR_EL1</a>	Debug Claim Tag Clear register
External	<a href="#">DBGCLAIMSET_EL1</a>	Debug Claim Tag Set register
External	<a href="#">DBGDTRRX_EL0</a>	Debug Data Transfer Register, Receive
External	<a href="#">DBGDTRTX_EL0</a>	Debug Data Transfer Register, Transmit
External	<a href="#">DBGWCR&lt;n&gt;_EL1</a>	Debug Watchpoint Control Registers
External	<a href="#">DBGWVR&lt;n&gt;_EL1</a>	Debug Watchpoint Value Registers
External	<a href="#">EDACR</a>	External Debug Auxiliary Control Register
External	<a href="#">EDCIDR0</a>	External Debug Component Identification Register 0
External	<a href="#">EDCIDR1</a>	External Debug Component Identification Register 1
External	<a href="#">EDCIDR2</a>	External Debug Component Identification Register 2
External	<a href="#">EDCIDR3</a>	External Debug Component Identification Register 3
External	<a href="#">EDCIDS</a>	External Debug Context ID Sample Register
External	<a href="#">EDDEVAFF0</a>	External Debug Device Affinity register 0
External	<a href="#">EDDEVAFF1</a>	External Debug Device Affinity register 1
External	<a href="#">EDDEVARCH</a>	External Debug Device Architecture register
External	<a href="#">EDDEVID</a>	External Debug Device ID register 0
External	<a href="#">EDDEVID1</a>	External Debug Device ID register 1
External	<a href="#">EDDEVID2</a>	External Debug Device ID register 2
External	<a href="#">EDDEVTYPE</a>	External Debug Device Type register
External	<a href="#">EDECCR</a>	External Debug Exception Catch Control Register
External	<a href="#">EDECR</a>	External Debug Execution Control Register
External	<a href="#">EDES</a>	External Debug Event Status Register
External	<a href="#">EDITCTRL</a>	External Debug Integration mode Control register
External	<a href="#">EDITR</a>	External Debug Instruction Transfer Register
External	<a href="#">EDLAR</a>	External Debug Lock Access Register
External	<a href="#">EDLSR</a>	External Debug Lock Status Register
External	<a href="#">EDPCSR</a>	External Debug Program Counter Sample Register
External	<a href="#">EDPIDR0</a>	External Debug Peripheral Identification Register 0
External	<a href="#">EDPIDR1</a>	External Debug Peripheral Identification Register 1
External	<a href="#">EDPIDR2</a>	External Debug Peripheral Identification Register 2
External	<a href="#">EDPIDR3</a>	External Debug Peripheral Identification Register 3
External	<a href="#">EDPIDR4</a>	External Debug Peripheral Identification Register 4

Exec state	Name	Description
External	<a href="#">EDPRCR</a>	External Debug Power/Reset Control Register
External	<a href="#">EDPRSR</a>	External Debug Processor Status Register
External	<a href="#">EDRCR</a>	External Debug Reserve Control Register
External	<a href="#">EDSCR</a>	External Debug Status and Control Register
External	<a href="#">EDVIDSR</a>	External Debug Virtual Context Sample Register
External	<a href="#">EDWAR</a>	External Debug Watchpoint Address Register
External	<a href="#">OSLAR_EL1</a>	OS Lock Access Register

## In the CTI functional group:

Exec state	Name	Description
External	<a href="#">ASICCTL</a>	CTI External Multiplexer Control register
External	<a href="#">CTIAPPCLEAR</a>	CTI Application Trigger Clear register
External	<a href="#">CTIAPPULSE</a>	CTI Application Pulse register
External	<a href="#">CTIAPPSET</a>	CTI Application Trigger Set register
External	<a href="#">CTIAUTHSTATUS</a>	CTI Authentication Status register
External	<a href="#">CTICHINSTATUS</a>	CTI Channel In Status register
External	<a href="#">CTICHOUTSTATUS</a>	CTI Channel Out Status register
External	<a href="#">CTICIDR0</a>	CTI Component Identification Register 0
External	<a href="#">CTICIDR1</a>	CTI Component Identification Register 1
External	<a href="#">CTICIDR2</a>	CTI Component Identification Register 2
External	<a href="#">CTICIDR3</a>	CTI Component Identification Register 3
External	<a href="#">CTICLAIMCLR</a>	CTI Claim Tag Clear register
External	<a href="#">CTICLAIMSET</a>	CTI Claim Tag Set register
External	<a href="#">CTICONTROL</a>	CTI Control register
External	<a href="#">CTIDEVAFF0</a>	CTI Device Affinity register 0
External	<a href="#">CTIDEVAFF1</a>	CTI Device Affinity register 1
External	<a href="#">CTIDEVARCH</a>	CTI Device Architecture register
External	<a href="#">CTIDEVID</a>	CTI Device ID register 0
External	<a href="#">CTIDEVID1</a>	CTI Device ID register 1
External	<a href="#">CTIDEVID2</a>	CTI Device ID register 2
External	<a href="#">CTIDEVTYPE</a>	CTI Device Type register
External	<a href="#">CTIGATE</a>	CTI Channel Gate Enable register
External	<a href="#">CTIINEN&lt;n&gt;</a>	CTI Input Trigger to Output Channel Enable registers
External	<a href="#">CTIINTACK</a>	CTI Output Trigger Acknowledge register
External	<a href="#">CTIITCTRL</a>	CTI Integration mode Control register
External	<a href="#">CTILAR</a>	CTI Lock Access Register
External	<a href="#">CTILSR</a>	CTI Lock Status Register
External	<a href="#">CTIOUTEN&lt;n&gt;</a>	CTI Input Channel to Output Trigger Enable registers
External	<a href="#">CTIPIDR0</a>	CTI Peripheral Identification Register 0
External	<a href="#">CTIPIDR1</a>	CTI Peripheral Identification Register 1
External	<a href="#">CTIPIDR2</a>	CTI Peripheral Identification Register 2
External	<a href="#">CTIPIDR3</a>	CTI Peripheral Identification Register 3
External	<a href="#">CTIPIDR4</a>	CTI Peripheral Identification Register 4
External	<a href="#">CTITRIGINSTATUS</a>	CTI Trigger In Status register
External	<a href="#">CTITRIGOUTSTATUS</a>	CTI Trigger Out Status register

## In the Virt functional group:

Exec state	Name	Description
AArch32	<a href="#">ATS1HR</a>	Address Translate Stage 1 Hyp mode Read
AArch32	<a href="#">ATS1HW</a>	Address Translate Stage 1 Hyp mode Write
AArch32	<a href="#">CNTHCTL</a>	Counter-timer Hyp Control register
AArch32	<a href="#">CNTHP_CVAL</a>	Counter-timer Hyp Physical CompareValue register
AArch32	<a href="#">CNTHP_TVAL</a>	Counter-timer Hyp Physical Timer TimerValue register
AArch32	<a href="#">CNTVOFF</a>	Counter-timer Virtual Offset register
AArch32	<a href="#">HACR</a>	Hyp Auxiliary Configuration Register
AArch32	<a href="#">HACTLR</a>	Hyp Auxiliary Control Register
AArch32	<a href="#">HACTLR2</a>	Hyp Auxiliary Control Register 2
AArch32	<a href="#">HADFSR</a>	Hyp Auxiliary Data Fault Status Register
AArch32	<a href="#">HAIFSR</a>	Hyp Auxiliary Instruction Fault Status Register
AArch32	<a href="#">HAMAIRO</a>	Hyp Auxiliary Memory Attribute Indirection Register 0

Exec state	Name	Description
AArch32	<a href="#">HMAAIR1</a>	Hyp Auxiliary Memory Attribute Indirection Register 1
AArch32	<a href="#">HCPTR</a>	Hyp Architectural Feature Trap Register
AArch32	<a href="#">HCR</a>	Hyp Configuration Register
AArch32	<a href="#">HCR2</a>	Hyp Configuration Register 2
AArch32	<a href="#">HDCR</a>	Hyp Debug Control Register
AArch32	<a href="#">HDFAR</a>	Hyp Data Fault Address Register
AArch32	<a href="#">HIFAR</a>	Hyp Instruction Fault Address Register
AArch32	<a href="#">HMAIR0</a>	Hyp Memory Attribute Indirection Register 0
AArch32	<a href="#">HMAIR1</a>	Hyp Memory Attribute Indirection Register 1
AArch32	<a href="#">HPFAR</a>	Hyp IPA Fault Address Register
AArch32	<a href="#">HRMR</a>	Hyp Reset Management Register
AArch32	<a href="#">HSCCLR</a>	Hyp System Control Register
AArch32	<a href="#">HSR</a>	Hyp Syndrome Register
AArch32	<a href="#">HSTR</a>	Hyp System Trap Register
AArch32	<a href="#">HTCR</a>	Hyp Translation Control Register
AArch32	<a href="#">HTPIDR</a>	Hyp Software Thread ID Register
AArch32	<a href="#">HTTBR</a>	Hyp Translation Table Base Register
AArch32	<a href="#">HVBAR</a>	Hyp Vector Base Address Register
AArch32	<a href="#">ICC_HSRE</a>	Interrupt Controller Hyp System Register Enable register
AArch32	<a href="#">ICH_AP0R&lt;n&gt;</a>	Interrupt Controller Hyp Active Priorities Group 0 Registers
AArch32	<a href="#">ICH_APIR&lt;n&gt;</a>	Interrupt Controller Hyp Active Priorities Group 1 Registers
AArch32	<a href="#">ICH_EISR</a>	Interrupt Controller End of Interrupt Status Register
AArch32	<a href="#">ICH_ELSR</a>	Interrupt Controller Empty List Register Status Register
AArch32	<a href="#">ICH_HCR</a>	Interrupt Controller Hyp Control Register
AArch32	<a href="#">ICH_LR&lt;n&gt;</a>	Interrupt Controller List Registers
AArch32	<a href="#">ICH_LRC&lt;n&gt;</a>	Interrupt Controller List Registers
AArch32	<a href="#">ICH_MISR</a>	Interrupt Controller Maintenance Interrupt State Register
AArch32	<a href="#">ICH_VMCR</a>	Interrupt Controller Virtual Machine Control Register
AArch32	<a href="#">ICH_VTR</a>	Interrupt Controller VGIC Type Register
AArch32	<a href="#">TLBIALLH</a>	TLB Invalidate All, Hyp mode
AArch32	<a href="#">TLBIALLHIS</a>	TLB Invalidate All, Hyp mode, Inner Shareable
AArch32	<a href="#">TLBIIPAS2</a>	TLB Invalidate by Intermediate Physical Address, Stage 2
AArch32	<a href="#">TLBIIPAS2IS</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Inner Shareable
AArch32	<a href="#">TLBIIPAS2L</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level
AArch32	<a href="#">TLBIIPAS2LIS</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, Inner Shareable
AArch32	<a href="#">TLBIMVAH</a>	TLB Invalidate by VA, Hyp mode
AArch32	<a href="#">TLBIMVAHIS</a>	TLB Invalidate by VA, Hyp mode, Inner Shareable
AArch32	<a href="#">TLBIMVALH</a>	TLB Invalidate by VA, Last level, Hyp mode
AArch32	<a href="#">TLBIMVALHIS</a>	TLB Invalidate by VA, Last level, Hyp mode, Inner Shareable
AArch32	<a href="#">VMPIDR</a>	Virtualization Multiprocessor ID Register
AArch32	<a href="#">VPIDR</a>	Virtualization Processor ID Register
AArch32	<a href="#">VTCR</a>	Virtualization Translation Control Register
AArch32	<a href="#">VTTBR</a>	Virtualization Translation Table Base Register
AArch64	<a href="#">ACTLR_EL2</a>	Auxiliary Control Register (EL2)
AArch64	<a href="#">AFSR0_EL2</a>	Auxiliary Fault Status Register 0 (EL2)
AArch64	<a href="#">AFSR1_EL2</a>	Auxiliary Fault Status Register 1 (EL2)
AArch64	<a href="#">AMAIR_EL2</a>	Auxiliary Memory Attribute Indirection Register (EL2)
AArch64	<a href="#">CNTHCTL_EL2</a>	Counter-timer Hypervisor Control register
AArch64	<a href="#">CNTHP_CTL_EL2</a>	Counter-timer Hypervisor Physical Timer Control register
AArch64	<a href="#">CNTHP_CVAL_EL2</a>	Counter-timer Hypervisor Physical Timer Compare Value register
AArch64	<a href="#">CNTHP_TVAL_EL2</a>	Counter-timer Hypervisor Physical Timer Timer Value register
AArch64	<a href="#">CNTVOFF_EL2</a>	Counter-timer Virtual Offset register
AArch64	<a href="#">CPTR_EL2</a>	Architectural Feature Trap Register (EL2)
AArch64	<a href="#">ESR_EL2</a>	Exception Syndrome Register (EL2)
AArch64	<a href="#">FAR_EL2</a>	Fault Address Register (EL2)
AArch64	<a href="#">HACR_EL2</a>	Hypervisor Auxiliary Control Register
AArch64	<a href="#">HCR_EL2</a>	Hypervisor Configuration Register
AArch64	<a href="#">HPFAR_EL2</a>	Hypervisor IPA Fault Address Register
AArch64	<a href="#">HSTR_EL2</a>	Hypervisor System Trap Register
AArch64	<a href="#">ICC_SRE_EL2</a>	Interrupt Controller System Register Enable register (EL2)
AArch64	<a href="#">ICH_AP0R&lt;n&gt;_EL2</a>	Interrupt Controller Hyp Active Priorities Group 0 Registers
AArch64	<a href="#">ICH_APIR&lt;n&gt;_EL2</a>	Interrupt Controller Hyp Active Priorities Group 1 Registers
AArch64	<a href="#">ICH_EISR_EL2</a>	Interrupt Controller End of Interrupt Status Register
AArch64	<a href="#">ICH_ELSR_EL2</a>	Interrupt Controller Empty List Register Status Register

Exec state	Name	Description
AArch64	<a href="#">ICH_HCR_EL2</a>	Interrupt Controller Hyp Control Register
AArch64	<a href="#">ICH_LR&lt;n&gt;_EL2</a>	Interrupt Controller List Registers
AArch64	<a href="#">ICH_MISR_EL2</a>	Interrupt Controller Maintenance Interrupt State Register
AArch64	<a href="#">ICH_VMCR_EL2</a>	Interrupt Controller Virtual Machine Control Register
AArch64	<a href="#">ICH_VTR_EL2</a>	Interrupt Controller VGIC Type Register
AArch64	<a href="#">TLBI_IPAS2E1</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1
AArch64	<a href="#">TLBI_IPAS2E1IS</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, EL1, Inner Shareable
AArch64	<a href="#">TLBI_IPAS2LE1</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1
AArch64	<a href="#">TLBI_IPAS2LE1IS</a>	TLB Invalidate by Intermediate Physical Address, Stage 2, Last level, EL1, Inner Shareable
AArch64	<a href="#">MAIR_EL2</a>	Memory Attribute Indirection Register (EL2)
AArch64	<a href="#">MDCR_EL2</a>	Monitor Debug Configuration Register (EL2)
AArch64	<a href="#">RMR_EL2</a>	Reset Management Register (EL2)
AArch64	<a href="#">SCTLR_EL2</a>	System Control Register (EL2)
AArch64	<a href="#">TCR_EL2</a>	Translation Control Register (EL2)
AArch64	<a href="#">TPIDR_EL2</a>	EL2 Software Thread ID Register
AArch64	<a href="#">TTBR0_EL2</a>	Translation Table Base Register 0 (EL2)
AArch64	<a href="#">TTBR1_EL2</a>	Translation Table Base Register 1 (EL2)
AArch64	<a href="#">VBAR_EL2</a>	Vector Base Address Register (EL2)
AArch64	<a href="#">VMPIDR_EL2</a>	Virtualization Multiprocessor ID Register
AArch64	<a href="#">VPIDR_EL2</a>	Virtualization Processor ID Register
AArch64	<a href="#">VTCR_EL2</a>	Virtualization Translation Control Register
AArch64	<a href="#">VTTBR_EL2</a>	Virtualization Translation Table Base Register

## In the Secure functional group:

Exec state	Name	Description
AArch32	<a href="#">ICC_MCTLR</a>	Interrupt Controller Monitor Control Register
AArch32	<a href="#">ICC_MSRE</a>	Interrupt Controller Monitor System Register Enable register
AArch32	<a href="#">MVBAR</a>	Monitor Vector Base Address Register
AArch32	<a href="#">NSACR</a>	Non-Secure Access Control Register
AArch32	<a href="#">SCR</a>	Secure Configuration Register
AArch32	<a href="#">SDCR</a>	Secure Debug Control Register
AArch32	<a href="#">SDER</a>	Secure Debug Enable Register
AArch64	<a href="#">ACTLR_EL3</a>	Auxiliary Control Register (EL3)
AArch64	<a href="#">AFSR0_EL3</a>	Auxiliary Fault Status Register 0 (EL3)
AArch64	<a href="#">AFSR1_EL3</a>	Auxiliary Fault Status Register 1 (EL3)
AArch64	<a href="#">AMAIR_EL3</a>	Auxiliary Memory Attribute Indirection Register (EL3)
AArch64	<a href="#">CPTR_EL3</a>	Architectural Feature Trap Register (EL3)
AArch64	<a href="#">ICC_CTLR_EL3</a>	Interrupt Controller Control Register (EL3)
AArch64	<a href="#">ICC_SRE_EL3</a>	Interrupt Controller System Register Enable register (EL3)
AArch64	<a href="#">MDCR_EL3</a>	Monitor Debug Configuration Register (EL3)
AArch64	<a href="#">SCR_EL3</a>	Secure Configuration Register
AArch64	<a href="#">SDER32_EL3</a>	AArch32 Secure Debug Enable Register
AArch64	<a href="#">VBAR_EL3</a>	Vector Base Address Register (EL3)

## In the Float functional group:

Exec state	Name	Description
AArch32	<a href="#">FPEXC</a>	Floating-Point Exception Control register
AArch32	<a href="#">FPSCR</a>	Floating-Point Status and Control Register
AArch32	<a href="#">FPSID</a>	Floating-Point System ID register
AArch32	<a href="#">MVFR0</a>	Media and VFP Feature Register 0
AArch32	<a href="#">MVFR1</a>	Media and VFP Feature Register 1
AArch32	<a href="#">MVFR2</a>	Media and VFP Feature Register 2
AArch64	<a href="#">FPCR</a>	Floating-point Control Register
AArch64	<a href="#">FPEXC32_EL2</a>	Floating-Point Exception Control register
AArch64	<a href="#">FPSR</a>	Floating-point Status Register
AArch64	<a href="#">MVFR0_EL1</a>	AArch32 Media and VFP Feature Register 0
AArch64	<a href="#">MVFR1_EL1</a>	AArch32 Media and VFP Feature Register 1
AArch64	<a href="#">MVFR2_EL1</a>	AArch32 Media and VFP Feature Register 2



## In the Legacy functional group:

Exec state	Name	Description
AArch32	<a href="#">CP15DMB</a>	Data Memory Barrier System instruction
AArch32	<a href="#">CP15DSB</a>	Data Synchronization Barrier System instruction
AArch32	<a href="#">CP15ISB</a>	Instruction Synchronization Barrier System instruction
AArch32	<a href="#">FCSEIDR</a>	FCSE Process ID register
AArch32	<a href="#">JIDR</a>	Jazelle ID Register
AArch32	<a href="#">JMCR</a>	Jazelle Main Configuration Register
AArch32	<a href="#">JOSCR</a>	Jazelle OS Control Register

## In the GIC functional group:

Exec state	Name	Description
AArch32	<a href="#">ICC_APOR&lt;n&gt;</a>	Interrupt Controller Active Priorities Group 0 Registers
AArch32	<a href="#">ICC_APIR&lt;n&gt;</a>	Interrupt Controller Active Priorities Group 1 Registers
AArch32	<a href="#">ICC_ASGI1R</a>	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
AArch32	<a href="#">ICC_BPR0</a>	Interrupt Controller Binary Point Register 0
AArch32	<a href="#">ICC_BPR1</a>	Interrupt Controller Binary Point Register 1
AArch32	<a href="#">ICC_CTLR</a>	Interrupt Controller Control Register
AArch32	<a href="#">ICC_DIR</a>	Interrupt Controller Deactivate Interrupt Register
AArch32	<a href="#">ICC_EOIR0</a>	Interrupt Controller End Of Interrupt Register 0
AArch32	<a href="#">ICC_EOIR1</a>	Interrupt Controller End Of Interrupt Register 1
AArch32	<a href="#">ICC_HPIR0</a>	Interrupt Controller Highest Priority Pending Interrupt Register 0
AArch32	<a href="#">ICC_HPIR1</a>	Interrupt Controller Highest Priority Pending Interrupt Register 1
AArch32	<a href="#">ICC_HSRE</a>	Interrupt Controller Hyp System Register Enable register
AArch32	<a href="#">ICC_IAR0</a>	Interrupt Controller Interrupt Acknowledge Register 0
AArch32	<a href="#">ICC_IAR1</a>	Interrupt Controller Interrupt Acknowledge Register 1
AArch32	<a href="#">ICC_IGRPEN0</a>	Interrupt Controller Interrupt Group 0 Enable register
AArch32	<a href="#">ICC_IGRPEN1</a>	Interrupt Controller Interrupt Group 1 Enable register
AArch32	<a href="#">ICC_MCTLR</a>	Interrupt Controller Monitor Control Register
AArch32	<a href="#">ICC_MGRPEN1</a>	Interrupt Controller Monitor Interrupt Group 1 Enable register
AArch32	<a href="#">ICC_MSRE</a>	Interrupt Controller Monitor System Register Enable register
AArch32	<a href="#">ICC_PMR</a>	Interrupt Controller Interrupt Priority Mask Register
AArch32	<a href="#">ICC_RPR</a>	Interrupt Controller Running Priority Register
AArch32	<a href="#">ICC_SGI0R</a>	Interrupt Controller Software Generated Interrupt Group 0 Register
AArch32	<a href="#">ICC_SGI1R</a>	Interrupt Controller Software Generated Interrupt Group 1 Register
AArch32	<a href="#">ICC_SRE</a>	Interrupt Controller System Register Enable register
AArch32	<a href="#">ICH_APOR&lt;n&gt;</a>	Interrupt Controller Hyp Active Priorities Group 0 Registers
AArch32	<a href="#">ICH_APIR&lt;n&gt;</a>	Interrupt Controller Hyp Active Priorities Group 1 Registers
AArch32	<a href="#">ICH_EISR</a>	Interrupt Controller End of Interrupt Status Register
AArch32	<a href="#">ICH_ELRSR</a>	Interrupt Controller Empty List Register Status Register
AArch32	<a href="#">ICH_HCR</a>	Interrupt Controller Hyp Control Register
AArch32	<a href="#">ICH_LR&lt;n&gt;</a>	Interrupt Controller List Registers
AArch32	<a href="#">ICH_LRC&lt;n&gt;</a>	Interrupt Controller List Registers
AArch32	<a href="#">ICH_MISR</a>	Interrupt Controller Maintenance Interrupt State Register
AArch32	<a href="#">ICH_VMCR</a>	Interrupt Controller Virtual Machine Control Register
AArch32	<a href="#">ICH_VTR</a>	Interrupt Controller VGIC Type Register
AArch32	<a href="#">ICV_APOR&lt;n&gt;</a>	Interrupt Controller Virtual Active Priorities Group 0 Registers
AArch32	<a href="#">ICV_APIR&lt;n&gt;</a>	Interrupt Controller Virtual Active Priorities Group 1 Registers
AArch32	<a href="#">ICV_BPR0</a>	Interrupt Controller Virtual Binary Point Register 0
AArch32	<a href="#">ICV_BPR1</a>	Interrupt Controller Virtual Binary Point Register 1
AArch32	<a href="#">ICV_CTLR</a>	Interrupt Controller Virtual Control Register
AArch32	<a href="#">ICV_DIR</a>	Interrupt Controller Deactivate Virtual Interrupt Register
AArch32	<a href="#">ICV_EOIR0</a>	Interrupt Controller Virtual End Of Interrupt Register 0
AArch32	<a href="#">ICV_EOIR1</a>	Interrupt Controller Virtual End Of Interrupt Register 1
AArch32	<a href="#">ICV_HPIR0</a>	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
AArch32	<a href="#">ICV_HPIR1</a>	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
AArch32	<a href="#">ICV_IAR0</a>	Interrupt Controller Virtual Interrupt Acknowledge Register 0
AArch32	<a href="#">ICV_IAR1</a>	Interrupt Controller Virtual Interrupt Acknowledge Register 1
AArch32	<a href="#">ICV_IGRPEN0</a>	Interrupt Controller Virtual Interrupt Group 0 Enable register
AArch32	<a href="#">ICV_IGRPEN1</a>	Interrupt Controller Virtual Interrupt Group 1 Enable register
AArch32	<a href="#">ICV_PMR</a>	Interrupt Controller Virtual Interrupt Priority Mask Register
AArch32	<a href="#">ICV_RPR</a>	Interrupt Controller Virtual Running Priority Register

Exec state	Name	Description
AArch64	<a href="#">ICC_AP0R&lt;n&gt;_EL1</a>	Interrupt Controller Active Priorities Group 0 Registers
AArch64	<a href="#">ICC_AP1R&lt;n&gt;_EL1</a>	Interrupt Controller Active Priorities Group 1 Registers
AArch64	<a href="#">ICC_ASGI1R_EL1</a>	Interrupt Controller Alias Software Generated Interrupt Group 1 Register
AArch64	<a href="#">ICC_BPR0_EL1</a>	Interrupt Controller Binary Point Register 0
AArch64	<a href="#">ICC_BPR1_EL1</a>	Interrupt Controller Binary Point Register 1
AArch64	<a href="#">ICC_CTLR_EL1</a>	Interrupt Controller Control Register (EL1)
AArch64	<a href="#">ICC_CTLR_EL3</a>	Interrupt Controller Control Register (EL3)
AArch64	<a href="#">ICC_DIR_EL1</a>	Interrupt Controller Deactivate Interrupt Register
AArch64	<a href="#">ICC_EOIR0_EL1</a>	Interrupt Controller End Of Interrupt Register 0
AArch64	<a href="#">ICC_EOIR1_EL1</a>	Interrupt Controller End Of Interrupt Register 1
AArch64	<a href="#">ICC_HPIR0_EL1</a>	Interrupt Controller Highest Priority Pending Interrupt Register 0
AArch64	<a href="#">ICC_HPIR1_EL1</a>	Interrupt Controller Highest Priority Pending Interrupt Register 1
AArch64	<a href="#">ICC_IAR0_EL1</a>	Interrupt Controller Interrupt Acknowledge Register 0
AArch64	<a href="#">ICC_IAR1_EL1</a>	Interrupt Controller Interrupt Acknowledge Register 1
AArch64	<a href="#">ICC_IGRPEN0_EL1</a>	Interrupt Controller Interrupt Group 0 Enable register
AArch64	<a href="#">ICC_IGRPEN1_EL1</a>	Interrupt Controller Interrupt Group 1 Enable register
AArch64	<a href="#">ICC_IGRPEN1_EL3</a>	Interrupt Controller Interrupt Group 1 Enable register (EL3)
AArch64	<a href="#">ICC_PMR_EL1</a>	Interrupt Controller Interrupt Priority Mask Register
AArch64	<a href="#">ICC_RPR_EL1</a>	Interrupt Controller Running Priority Register
AArch64	<a href="#">ICC_SGI0R_EL1</a>	Interrupt Controller Software Generated Interrupt Group 0 Register
AArch64	<a href="#">ICC_SGI1R_EL1</a>	Interrupt Controller Software Generated Interrupt Group 1 Register
AArch64	<a href="#">ICC_SRE_EL1</a>	Interrupt Controller System Register Enable register (EL1)
AArch64	<a href="#">ICC_SRE_EL2</a>	Interrupt Controller System Register Enable register (EL2)
AArch64	<a href="#">ICC_SRE_EL3</a>	Interrupt Controller System Register Enable register (EL3)
AArch64	<a href="#">ICH_AP0R&lt;n&gt;_EL2</a>	Interrupt Controller Hyp Active Priorities Group 0 Registers
AArch64	<a href="#">ICH_AP1R&lt;n&gt;_EL2</a>	Interrupt Controller Hyp Active Priorities Group 1 Registers
AArch64	<a href="#">ICH_EISR_EL2</a>	Interrupt Controller End of Interrupt Status Register
AArch64	<a href="#">ICH_ELRSR_EL2</a>	Interrupt Controller Empty List Register Status Register
AArch64	<a href="#">ICH_HCR_EL2</a>	Interrupt Controller Hyp Control Register
AArch64	<a href="#">ICH_LR&lt;n&gt;_EL2</a>	Interrupt Controller List Registers
AArch64	<a href="#">ICH_MISR_EL2</a>	Interrupt Controller Maintenance Interrupt State Register
AArch64	<a href="#">ICH_VMCR_EL2</a>	Interrupt Controller Virtual Machine Control Register
AArch64	<a href="#">ICH_VTR_EL2</a>	Interrupt Controller VGIC Type Register
AArch64	<a href="#">ICV_AP0R&lt;n&gt;_EL1</a>	Interrupt Controller Virtual Active Priorities Group 0 Registers
AArch64	<a href="#">ICV_AP1R&lt;n&gt;_EL1</a>	Interrupt Controller Virtual Active Priorities Group 1 Registers
AArch64	<a href="#">ICV_BPR0_EL1</a>	Interrupt Controller Virtual Binary Point Register 0
AArch64	<a href="#">ICV_BPR1_EL1</a>	Interrupt Controller Virtual Binary Point Register 1
AArch64	<a href="#">ICV_CTLR_EL1</a>	Interrupt Controller Virtual Control Register
AArch64	<a href="#">ICV_DIR_EL1</a>	Interrupt Controller Deactivate Virtual Interrupt Register
AArch64	<a href="#">ICV_EOIR0_EL1</a>	Interrupt Controller Virtual End Of Interrupt Register 0
AArch64	<a href="#">ICV_EOIR1_EL1</a>	Interrupt Controller Virtual End Of Interrupt Register 1
AArch64	<a href="#">ICV_HPIR0_EL1</a>	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 0
AArch64	<a href="#">ICV_HPIR1_EL1</a>	Interrupt Controller Virtual Highest Priority Pending Interrupt Register 1
AArch64	<a href="#">ICV_IAR0_EL1</a>	Interrupt Controller Virtual Interrupt Acknowledge Register 0
AArch64	<a href="#">ICV_IAR1_EL1</a>	Interrupt Controller Virtual Interrupt Acknowledge Register 1
AArch64	<a href="#">ICV_IGRPEN0_EL1</a>	Interrupt Controller Virtual Interrupt Group 0 Enable register
AArch64	<a href="#">ICV_IGRPEN1_EL1</a>	Interrupt Controller Virtual Interrupt Group 1 Enable register
AArch64	<a href="#">ICV_PMR_EL1</a>	Interrupt Controller Virtual Interrupt Priority Mask Register
AArch64	<a href="#">ICV_RPR_EL1</a>	Interrupt Controller Virtual Running Priority Register

## In the GICD functional group:

Exec state	Name	Description
External	<a href="#">GICD_CLRSPI_NSR</a>	Clear Non-secure SPI Pending Register
External	<a href="#">GICD_CLRSPI_SR</a>	Clear Secure SPI Pending Register
External	<a href="#">GICD_CPENDSGIR&lt;n&gt;</a>	SPI Clear-Pending Registers
External	<a href="#">GICD_CTLR</a>	Distributor Control Register
External	<a href="#">GICD_ICACTIVER&lt;n&gt;</a>	Interrupt Clear-Active Registers
External	<a href="#">GICD_ICENABLER&lt;n&gt;</a>	Interrupt Clear-Enable Registers
External	<a href="#">GICD_ICFGR&lt;n&gt;</a>	Interrupt Configuration Registers
External	<a href="#">GICD_ICPENDR&lt;n&gt;</a>	Interrupt Clear-Pending Registers
External	<a href="#">GICD_IGROUPR&lt;n&gt;</a>	Interrupt Group Registers
External	<a href="#">GICD_IGRPMODR&lt;n&gt;</a>	Interrupt Group Modifier Registers

Exec state	Name	Description
External	<a href="#">GICD_IIDR</a>	Distributor Implementer Identification Register
External	<a href="#">GICD_IPRIORITYR&lt;n&gt;</a>	Interrupt Priority Registers
External	<a href="#">GICD_IROUTER&lt;n&gt;</a>	Interrupt Routing Registers
External	<a href="#">GICD_ISACTIVER&lt;n&gt;</a>	Interrupt Set-Active Registers
External	<a href="#">GICD_ISENBALER&lt;n&gt;</a>	Interrupt Set-Enable Registers
External	<a href="#">GICD_ISPENDR&lt;n&gt;</a>	Interrupt Set-Pending Registers
External	<a href="#">GICD_ITARGETSR&lt;n&gt;</a>	Interrupt Processor Targets Registers
External	<a href="#">GICD_NSACR&lt;n&gt;</a>	Non-secure Access Control Registers
External	<a href="#">GICD_SETSPI_NSR</a>	Set Non-secure SPI Pending Register
External	<a href="#">GICD_SETSPI_SR</a>	Set Secure SPI Pending Register
External	<a href="#">GICD_SGIR</a>	Software Generated Interrupt Register
External	<a href="#">GICD_SPENDSGIR&lt;n&gt;</a>	SGI Set-Pending Registers
External	<a href="#">GICD_STATUSR</a>	Error Reporting Status Register
External	<a href="#">GICD_TYPER</a>	Interrupt Controller Type Register

## In the GICR functional group:

Exec state	Name	Description
External	<a href="#">GICR_CLRLPIR</a>	Clear LPI Pending Register
External	<a href="#">GICR_CTLR</a>	Redistributor Control Register
External	<a href="#">GICR_ICACTIVER0</a>	Interrupt Clear-Active Register 0
External	<a href="#">GICR_ICENABLER0</a>	Interrupt Clear-Enable Register 0
External	<a href="#">GICR_ICFGR0</a>	Interrupt Configuration Register 0
External	<a href="#">GICR_ICFGR1</a>	Interrupt Configuration Register 1
External	<a href="#">GICR_ICPENDR0</a>	Interrupt Clear-Pending Register 0
External	<a href="#">GICR_IGROUPR0</a>	Interrupt Group Register 0
External	<a href="#">GICR_IGRPMODR0</a>	Interrupt Group Modifier Register 0
External	<a href="#">GICR_IIDR</a>	Redistributor Implementer Identification Register
External	<a href="#">GICR_INVALIDR</a>	Redistributor Invalidate All Register
External	<a href="#">GICR_INVLPIR</a>	Redistributor Invalidate LPI Register
External	<a href="#">GICR_IPRIORITYR&lt;n&gt;</a>	Interrupt Priority Registers
External	<a href="#">GICR_ISACTIVER0</a>	Interrupt Set-Active Register 0
External	<a href="#">GICR_ISENBALER0</a>	Interrupt Set-Enable Register 0
External	<a href="#">GICR_ISPENDR0</a>	Interrupt Set-Pending Register 0
External	<a href="#">GICR_NSACR</a>	Non-secure Access Control Register
External	<a href="#">GICR_PENDBASER</a>	Redistributor LPI Pending Table Base Address Register
External	<a href="#">GICR_PROPBASER</a>	Redistributor Properties Base Address Register
External	<a href="#">GICR_SETLPIR</a>	Set LPI Pending Register
External	<a href="#">GICR_STATUSR</a>	Error Reporting Status Register
External	<a href="#">GICR_SYNCR</a>	Redistributor Synchronize Register
External	<a href="#">GICR_TYPER</a>	Redistributor Type Register
External	<a href="#">GICR_VPENDBASER</a>	Virtual Redistributor LPI Pending Table Base Address Register
External	<a href="#">GICR_VPROPBASER</a>	Virtual Redistributor Properties Base Address Register
External	<a href="#">GICR_WAKER</a>	Redistributor Wake Register

## In the GICC functional group:

Exec state	Name	Description
External	<a href="#">GICC_ABPR</a>	CPU Interface Aliased Binary Point Register
External	<a href="#">GICC_AEOIR</a>	CPU Interface Aliased End Of Interrupt Register
External	<a href="#">GICC_AHPPIR</a>	CPU Interface Aliased Highest Priority Pending Interrupt Register
External	<a href="#">GICC_AIAR</a>	CPU Interface Aliased Interrupt Acknowledge Register
External	<a href="#">GICC_APR&lt;n&gt;</a>	CPU Interface Active Priorities Registers
External	<a href="#">GICC_BPR</a>	CPU Interface Binary Point Register
External	<a href="#">GICC_CTLR</a>	CPU Interface Control Register
External	<a href="#">GICC_DIR</a>	CPU Interface Deactivate Interrupt Register
External	<a href="#">GICC_EOIR</a>	CPU Interface End Of Interrupt Register
External	<a href="#">GICC_HPPIR</a>	CPU Interface Highest Priority Pending Interrupt Register
External	<a href="#">GICC_IAR</a>	CPU Interface Interrupt Acknowledge Register
External	<a href="#">GICC_IIDR</a>	CPU Interface Identification Register
External	<a href="#">GICC_NSAPR&lt;n&gt;</a>	CPU Interface Non-secure Active Priorities Registers
External	<a href="#">GICC_PMR</a>	CPU Interface Priority Mask Register



Exec state	Name	Description
External	<a href="#">GICC_RPR</a>	CPU Interface Running Priority Register
External	<a href="#">GICC_STATUSR</a>	CPU Interface Status Register

## In the GICV functional group:

Exec state	Name	Description
External	<a href="#">GICV_ABPR</a>	Virtual Machine Aliased Binary Point Register
External	<a href="#">GICV_AEOIR</a>	Virtual Machine Aliased End Of Interrupt Register
External	<a href="#">GICV_AHPPIR</a>	Virtual Machine Aliased Highest Priority Pending Interrupt Register
External	<a href="#">GICV_AIAR</a>	Virtual Machine Aliased Interrupt Acknowledge Register
External	<a href="#">GICV_APR&lt;n&gt;</a>	Virtual Machine Active Priorities Registers
External	<a href="#">GICV_BPR</a>	Virtual Machine Binary Point Register
External	<a href="#">GICV_CTLR</a>	Virtual Machine Control Register
External	<a href="#">GICV_DIR</a>	Virtual Machine Deactivate Interrupt Register
External	<a href="#">GICV_EOIR</a>	Virtual Machine End Of Interrupt Register
External	<a href="#">GICV_HPPIR</a>	Virtual Machine Highest Priority Pending Interrupt Register
External	<a href="#">GICV_IAR</a>	Virtual Machine Interrupt Acknowledge Register
External	<a href="#">GICV_IIDR</a>	Virtual Machine CPU Interface Identification Register
External	<a href="#">GICV_PMR</a>	Virtual Machine Priority Mask Register
External	<a href="#">GICV_RPR</a>	Virtual Machine Running Priority Register
External	<a href="#">GICV_STATUSR</a>	Virtual Machine Error Reporting Status Register

## In the GICH functional group:

Exec state	Name	Description
External	<a href="#">GICH_APR&lt;n&gt;</a>	Active Priorities Registers
External	<a href="#">GICH_EISR</a>	End Interrupt Status Register
External	<a href="#">GICH_ELRSR</a>	Empty List Register Status Register
External	<a href="#">GICH_HCR</a>	Hypervisor Control Register
External	<a href="#">GICH_LR&lt;n&gt;</a>	List Registers
External	<a href="#">GICH_MISR</a>	Maintenance Interrupt Status Register
External	<a href="#">GICH_VMCR</a>	Virtual Machine Control Register
External	<a href="#">GICH_VTR</a>	Virtual Type Register

## In the GITS functional group:

Exec state	Name	Description
External	<a href="#">GITS_BASER&lt;n&gt;</a>	ITS Translation Table Descriptors
External	<a href="#">GITS_CBASER</a>	ITS Command Queue Descriptor
External	<a href="#">GITS_CREADR</a>	ITS Read Register
External	<a href="#">GITS_CTLR</a>	ITS Control Register
External	<a href="#">GITS_CWRITER</a>	ITS Write Register
External	<a href="#">GITS_IIDR</a>	ITS Identification Register
External	<a href="#">GITS_TRANSLATER</a>	ITS Translation Register
External	<a href="#">GITS_TYPER</a>	ITS Type Register

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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

[CNTELOACR](#): Counter-timer EL0 Access Control Register

[CNTFID0](#): Counter Frequency ID

[CNTFID<n>](#): Counter Frequency IDs

[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

[CTIAPPULSE](#): CTI Application Pulse register

[CTIAPPSET](#): CTI Application Trigger Set register

[CTIAUTHSTATUS](#): CTI Authentication Status register

[CTICHINSTATUS](#): CTI Channel In Status register

[CTICHOUTSTATUS](#): CTI Channel Out Status register

[CTICIDR0](#): CTI Component Identification Register 0

[CTICIDR1](#): CTI Component Identification Register 1

[CTICIDR2](#): CTI Component Identification Register 2

[CTICIDR3](#): CTI Component Identification Register 3

[CTICLAIMCLR](#): CTI Claim Tag Clear register

[CTICLAIMSET](#): CTI Claim Tag Set register

[CTICONTROL](#): CTI Control register

[CTIDEVAFF0](#): CTI Device Affinity register 0

[CTIDEVAFF1](#): CTI Device Affinity register 1

[CTIDEVARCH](#): CTI Device Architecture register

[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

[EDECRCR](#): External Debug Execution Control Register

[EDESRCR](#): 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\\_HPPIR](#): CPU Interface Highest Priority Pending Interrupt Register  
[GICC\\_IAR](#): CPU Interface Interrupt Acknowledge Register  
[GICC\\_IIDR](#): CPU Interface Identification Register  
[GICC\\_NSAPR<n>](#): CPU Interface Non-secure Active Priorities Registers  
[GICC\\_PMR](#): CPU Interface Priority Mask Register  
[GICC\\_RPR](#): CPU Interface Running Priority Register  
[GICC\\_STATUSR](#): CPU Interface Status Register  
[GICD\\_CLRSPI\\_NSR](#): Clear Non-secure SPI Pending Register  
[GICD\\_CLRSPI\\_SR](#): Clear Secure SPI Pending Register  
[GICD\\_CPENDSGIR<n>](#): SGI Clear-Pending Registers  
[GICD\\_CTLR](#): Distributor Control Register  
[GICD\\_ICACTIVER<n>](#): Interrupt Clear-Active Registers  
[GICD\\_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

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# External register index by offset

Below are indexes for external registers in the following blocks:

- [PMU](#)
- [GIC CPU interface](#)
- [GIC Virtual interface control](#)
- [Timer](#)
- [Debug](#)
- [GIC Redistributor](#)
- [GIC Virtual CPU interface](#)
- [GIC ITS control](#)
- [GIC ITS translation](#)
- [CTI](#)
- [GIC Distributor](#)

## In the PMU block:

Offset	Name	Description
$0 \times 000 + 8n$	<a href="#">PMEVCNTR&lt;n&gt;_EL0</a>	Performance Monitors Event Count Registers
$0 \times 0F8$	<a href="#">PMCCNTR_EL0[31:0]</a>	Performance Monitors Cycle Counter
$0 \times 0FC$	<a href="#">PMCCNTR_EL0[63:32]</a>	Performance Monitors Cycle Counter
$0 \times 200$	<a href="#">PMPCSR[31:0]</a>	Program Counter Sample Register
$0 \times 204$	<a href="#">PMPCSR[63:32]</a>	Program Counter Sample Register
$0 \times 208$	<a href="#">PMCID1SR</a>	CONTEXTIDR_EL1 Sample Register
$0 \times 20C$	<a href="#">PMVIDSR</a>	VMID Sample Register
$0 \times 220$	<a href="#">PMPCSR[31:0]</a>	Program Counter Sample Register
$0 \times 224$	<a href="#">PMPCSR[63:32]</a>	Program Counter Sample Register
$0 \times 228$	<a href="#">PMCID1SR</a>	CONTEXTIDR_EL1 Sample Register
$0 \times 22C$	<a href="#">PMCID2SR</a>	CONTEXTIDR_EL2 Sample Register
$0 \times 400 + 4n$	<a href="#">PMEVTYPEPER&lt;n&gt;_EL0</a>	Performance Monitors Event Type Registers
$0 \times 47C$	<a href="#">PMCCFILTR_EL0</a>	Performance Monitors Cycle Counter Filter Register
$0 \times C00$	<a href="#">PMCNTENSET_EL0</a>	Performance Monitors Count Enable Set register
$0 \times C20$	<a href="#">PMCNTENCLR_EL0</a>	Performance Monitors Count Enable Clear register
$0 \times C40$	<a href="#">PMINTENSET_EL1</a>	Performance Monitors Interrupt Enable Set register
$0 \times C60$	<a href="#">PMINTENCLR_EL1</a>	Performance Monitors Interrupt Enable Clear register
$0 \times C80$	<a href="#">PMOVSCLR_EL0</a>	Performance Monitors Overflow Flag Status Clear register
$0 \times CA0$	<a href="#">PMSWINC_EL0</a>	Performance Monitors Software Increment register
$0 \times CC0$	<a href="#">PMOVSSET_EL0</a>	Performance Monitors Overflow Flag Status Set register
$0 \times E00$	<a href="#">PMCFGR</a>	Performance Monitors Configuration Register
$0 \times E04$	<a href="#">PMCR_EL0</a>	Performance Monitors Control Register
$0 \times E20$	<a href="#">PMCEID0</a>	Performance Monitors Common Event Identification register 0
$0 \times E24$	<a href="#">PMCEID1</a>	Performance Monitors Common Event Identification register 1
$0 \times E28$	<a href="#">PMCEID2</a>	Performance Monitors Common Event Identification register 2
$0 \times E2C$	<a href="#">PMCEID3</a>	Performance Monitors Common Event Identification register 3
$0 \times F00$	<a href="#">PMITCTRL</a>	Performance Monitors Integration mode Control register
$0 \times FA8$	<a href="#">PMDEVAFF0</a>	Performance Monitors Device Affinity register 0
$0 \times FAC$	<a href="#">PMDEVAFF1</a>	Performance Monitors Device Affinity register 1
$0 \times FB0$	<a href="#">PMLAR</a>	Performance Monitors Lock Access Register
$0 \times FB4$	<a href="#">PMLSR</a>	Performance Monitors Lock Status Register
$0 \times FB8$	<a href="#">PMAUTHSTATUS</a>	Performance Monitors Authentication Status register
$0 \times FBC$	<a href="#">PMDEVARCH</a>	Performance Monitors Device Architecture register

Offset	Name	Description
0xFC8	<a href="#">PMDEVID</a>	Performance Monitors Device ID register
0xFCC	<a href="#">PMDEVTYPE</a>	Performance Monitors Device Type register
0xFD0	<a href="#">PMPIDR4</a>	Performance Monitors Peripheral Identification Register 4
0xFE0	<a href="#">PMPIDR0</a>	Performance Monitors Peripheral Identification Register 0
0xFE4	<a href="#">PMPIDR1</a>	Performance Monitors Peripheral Identification Register 1
0xFE8	<a href="#">PMPIDR2</a>	Performance Monitors Peripheral Identification Register 2
0xFEC	<a href="#">PMPIDR3</a>	Performance Monitors Peripheral Identification Register 3
0xFF0	<a href="#">PMCIDR0</a>	Performance Monitors Component Identification Register 0
0xFF4	<a href="#">PMCIDR1</a>	Performance Monitors Component Identification Register 1
0xFF8	<a href="#">PMCIDR2</a>	Performance Monitors Component Identification Register 2
0xFFC	<a href="#">PMCIDR3</a>	Performance Monitors Component Identification Register 3

### In the GIC CPU interface block:

Offset	Name	Description
0x0000	<a href="#">GICC_CTLR</a>	CPU Interface Control Register
0x0004	<a href="#">GICC_PMR</a>	CPU Interface Priority Mask Register
0x0008	<a href="#">GICC_BPR</a>	CPU Interface Binary Point Register
0x000C	<a href="#">GICC_IAR</a>	CPU Interface Interrupt Acknowledge Register
0x0010	<a href="#">GICC_EOIR</a>	CPU Interface End Of Interrupt Register
0x0014	<a href="#">GICC_RPR</a>	CPU Interface Running Priority Register
0x0018	<a href="#">GICC_HPPIR</a>	CPU Interface Highest Priority Pending Interrupt Register
0x001C	<a href="#">GICC_ABPR</a>	CPU Interface Aliased Binary Point Register
0x0020–0x003C	<a href="#">GICC_AIAR</a>	CPU Interface Aliased Interrupt Acknowledge Register
0x0024	<a href="#">GICC_AEOIR</a>	CPU Interface Aliased End Of Interrupt Register
0x0028	<a href="#">GICC_AHPPIR</a>	CPU Interface Aliased Highest Priority Pending Interrupt Register
0x002C	<a href="#">GICC_STATUSR</a>	CPU Interface Status Register
0x00D0 + 4n	<a href="#">GICC_APR&lt;n&gt;</a>	CPU Interface Active Priorities Registers
0x00E0 + 4n	<a href="#">GICC_NSAPR&lt;n&gt;</a>	CPU Interface Non-secure Active Priorities Registers
0x00FC	<a href="#">GICC_IIDR</a>	CPU Interface Identification Register
0x1000	<a href="#">GICC_DIR</a>	CPU Interface Deactivate Interrupt Register

### In the GIC Virtual interface control block:

Offset	Name	Description
0x0000	<a href="#">GICH_HCR</a>	Hypervisor Control Register
0x0004	<a href="#">GICH_VTR</a>	Virtual Type Register
0x0008	<a href="#">GICH_VMCR</a>	Virtual Machine Control Register
0x0010	<a href="#">GICH_MISR</a>	Maintenance Interrupt Status Register
0x0020	<a href="#">GICH_EISR</a>	End Interrupt Status Register
0x0030	<a href="#">GICH_ELRSR</a>	Empty List Register Status Register
0x00F0 + 4n	<a href="#">GICH_APR&lt;n&gt;</a>	Active Priorities Registers
0x0100 + 4n	<a href="#">GICH_LR&lt;n&gt;</a>	List Registers

### In the Timer block:

Frame	Offset	Name	Description
CNTControlBase	0x000	<a href="#">CNTCR</a>	Counter Control Register
CNTControlBase	0x004	<a href="#">CNTSR</a>	Counter Status Register

Frame	Offset	Name	Description
CNTControlBase	0x008	<a href="#">CNTCV[31:0]</a>	Counter Count Value register
CNTControlBase	0x00C	<a href="#">CNTCV[63:32]</a>	Counter Count Value register
CNTControlBase	0x020	<a href="#">CNTFID0</a>	Counter Frequency ID
CNTControlBase	0x020 + 4n	<a href="#">CNTFID&lt;n&gt;</a>	Counter Frequency IDs
CNTControlBase	0xFD0 + 4n	<a href="#">CounterID&lt;n&gt;</a>	Counter ID registers
CNTReadBase	0x000	<a href="#">CNTCV[31:0]</a>	Counter Count Value register
CNTReadBase	0x004	<a href="#">CNTCV[63:32]</a>	Counter Count Value register
CNTReadBase	0xFD0 + 4n	<a href="#">CounterID&lt;n&gt;</a>	Counter ID registers
CNTBaseN	0x000	<a href="#">CNTPCT[31:0]</a>	Counter-timer Physical Count
CNTBaseN	0x004	<a href="#">CNTPCT[63:32]</a>	Counter-timer Physical Count
CNTBaseN	0x008	<a href="#">CNTVCT[31:0]</a>	Counter-timer Virtual Count
CNTBaseN	0x00C	<a href="#">CNTVCT[63:32]</a>	Counter-timer Virtual Count
CNTBaseN	0x010	<a href="#">CNTFRQ</a>	Counter-timer Frequency
CNTBaseN	0x014	<a href="#">CNTEL0ACR</a>	Counter-timer EL0 Access Control Register
CNTBaseN	0x018	<a href="#">CNTVOFF[31:0]</a>	Counter-timer Virtual Offset
CNTBaseN	0x01C	<a href="#">CNTVOFF[63:32]</a>	Counter-timer Virtual Offset
CNTBaseN	0x020	<a href="#">CNTP_CVAL[31:0]</a>	Counter-timer Physical Timer CompareValue
CNTBaseN	0x024	<a href="#">CNTP_CVAL[63:32]</a>	Counter-timer Physical Timer CompareValue
CNTBaseN	0x028	<a href="#">CNTP_TVAL</a>	Counter-timer Physical Timer TimerValue
CNTBaseN	0x02C	<a href="#">CNTP_CTL</a>	Counter-timer Physical Timer Control
CNTBaseN	0x030	<a href="#">CNTV_CVAL[31:0]</a>	Counter-timer Virtual Timer CompareValue
CNTBaseN	0x034	<a href="#">CNTV_CVAL[63:32]</a>	Counter-timer Virtual Timer CompareValue
CNTBaseN	0x038	<a href="#">CNTV_TVAL</a>	Counter-timer Virtual Timer TimerValue
CNTBaseN	0x03C	<a href="#">CNTV_CTL</a>	Counter-timer Virtual Timer Control
CNTBaseN	0xFD0 + 4n	<a href="#">CounterID&lt;n&gt;</a>	Counter ID registers
CNTEL0BaseN	0x000	<a href="#">CNTPCT[31:0]</a>	Counter-timer Physical Count
CNTEL0BaseN	0x004	<a href="#">CNTPCT[63:32]</a>	Counter-timer Physical Count
CNTEL0BaseN	0x008	<a href="#">CNTVCT[31:0]</a>	Counter-timer Virtual Count
CNTEL0BaseN	0x00C	<a href="#">CNTVCT[63:32]</a>	Counter-timer Virtual Count
CNTEL0BaseN	0x010	<a href="#">CNTFRQ</a>	Counter-timer Frequency
CNTEL0BaseN	0x020	<a href="#">CNTP_CVAL[31:0]</a>	Counter-timer Physical Timer CompareValue
CNTEL0BaseN	0x024	<a href="#">CNTP_CVAL[63:32]</a>	Counter-timer Physical Timer CompareValue
CNTEL0BaseN	0x028	<a href="#">CNTP_TVAL</a>	Counter-timer Physical Timer TimerValue
CNTEL0BaseN	0x02C	<a href="#">CNTP_CTL</a>	Counter-timer Physical Timer Control
CNTEL0BaseN	0x030	<a href="#">CNTV_CVAL[31:0]</a>	Counter-timer Virtual Timer CompareValue
CNTEL0BaseN	0x034	<a href="#">CNTV_CVAL[63:32]</a>	Counter-timer Virtual Timer CompareValue
CNTEL0BaseN	0x038	<a href="#">CNTV_TVAL</a>	Counter-timer Virtual Timer TimerValue
CNTEL0BaseN	0x03C	<a href="#">CNTV_CTL</a>	Counter-timer Virtual Timer Control
CNTEL0BaseN	0xFD0 + 4n	<a href="#">CounterID&lt;n&gt;</a>	Counter ID registers
CNTCTLLBase	0x000	<a href="#">CNTFRQ</a>	Counter-timer Frequency
CNTCTLLBase	0x004	<a href="#">CNTNSAR</a>	Counter-timer Non-secure Access Register
CNTCTLLBase	0x008	<a href="#">CNTTIDR</a>	Counter-timer Timer ID Register
CNTCTLLBase	0x040 + 4n	<a href="#">CNTACR&lt;n&gt;</a>	Counter-timer Access Control Registers
CNTCTLLBase	0x080 + 8n	<a href="#">CNTVOFF&lt;n&gt;[31:0]</a>	Counter-timer Virtual Offsets
CNTCTLLBase	0x084 + 8n	<a href="#">CNTVOFF&lt;n&gt;[63:32]</a>	Counter-timer Virtual Offsets
CNTCTLLBase	0xFD0 + 4n	<a href="#">CounterID&lt;n&gt;</a>	Counter ID registers

## In the Debug block:

Offset	Name	Description
0x020	<a href="#">EDES</a>	External Debug Event Status Register
0x024	<a href="#">EDECR</a>	External Debug Execution Control Register
0x030	<a href="#">EDWAR[31:0]</a>	External Debug Watchpoint Address Register
0x034	<a href="#">EDWAR[63:32]</a>	External Debug Watchpoint Address Register
0x080	<a href="#">DBGDTRRX_EL0</a>	Debug Data Transfer Register, Receive
0x084	<a href="#">EDITR</a>	External Debug Instruction Transfer Register
0x088	<a href="#">EDSCR</a>	External Debug Status and Control Register
0x08C	<a href="#">DBGDTRTX_EL0</a>	Debug Data Transfer Register, Transmit
0x090	<a href="#">EDRCR</a>	External Debug Reserve Control Register
0x094	<a href="#">EDACR</a>	External Debug Auxiliary Control Register
0x098	<a href="#">EDECCR</a>	External Debug Exception Catch Control Register
0x0A0	<a href="#">EDPCSR[31:0]</a>	External Debug Program Counter Sample Register
0x0A4	<a href="#">EDCIDS</a>	External Debug Context ID Sample Register
0x0A8	<a href="#">EDVIDSR</a>	External Debug Virtual Context Sample Register
0x0AC	<a href="#">EDPCSR[63:32]</a>	External Debug Program Counter Sample Register
0x300	<a href="#">OSLAR_EL1</a>	OS Lock Access Register
0x310	<a href="#">EDPRCR</a>	External Debug Power/Reset Control Register
0x314	<a href="#">EDPRSR</a>	External Debug Processor Status Register
0x400 + 16n	<a href="#">DBGBVR&lt;n&gt;_EL1[31:0]</a>	Debug Breakpoint Value Registers
0x404 + 16n	<a href="#">DBGBVR&lt;n&gt;_EL1[63:32]</a>	Debug Breakpoint Value Registers
0x408 + 16n	<a href="#">DBGBCR&lt;n&gt;_EL1</a>	Debug Breakpoint Control Registers
0x800 + 16n	<a href="#">DBGWVR&lt;n&gt;_EL1[31:0]</a>	Debug Watchpoint Value Registers
0x804 + 16n	<a href="#">DBGWVR&lt;n&gt;_EL1[63:32]</a>	Debug Watchpoint Value Registers
0x808 + 16n	<a href="#">DBGWCR&lt;n&gt;_EL1</a>	Debug Watchpoint Control Registers
0xD00	<a href="#">MIDR_EL1</a>	Main ID Register
0xD20	<a href="#">EDPFR[31:0]</a>	External Debug Processor Feature Register
0xD24	<a href="#">EDPFR[63:32]</a>	External Debug Processor Feature Register
0xD28	<a href="#">EDDFR[31:0]</a>	External Debug Feature Register
0xD2C	<a href="#">EDDFR[63:32]</a>	External Debug Feature Register
0xD60	<a href="#">EDAA32PFR</a>	External Debug AArch32 Processor Feature Register
0xF00	<a href="#">EDITCTRL</a>	External Debug Integration mode Control register
0xFA0	<a href="#">DBGCLAIMSET_EL1</a>	Debug Claim Tag Set register
0xFA4	<a href="#">DBGCLAIMCLR_EL1</a>	Debug Claim Tag Clear register
0xFA8	<a href="#">EDDEVAFF0</a>	External Debug Device Affinity register 0
0xFAC	<a href="#">EDDEVAFF1</a>	External Debug Device Affinity register 1
0xFB0	<a href="#">EDLAR</a>	External Debug Lock Access Register
0xFB4	<a href="#">EDLSR</a>	External Debug Lock Status Register
0xFB8	<a href="#">DBGAUTHSTATUS_EL1</a>	Debug Authentication Status register
0xFBC	<a href="#">EDDEVARCH</a>	External Debug Device Architecture register
0xFC0	<a href="#">EDDEVID2</a>	External Debug Device ID register 2
0xFC4	<a href="#">EDDEVID1</a>	External Debug Device ID register 1
0xFC8	<a href="#">EDDEVID</a>	External Debug Device ID register 0
0xFCC	<a href="#">EDDEVTYPE</a>	External Debug Device Type register
0xFD0	<a href="#">EDPIDR4</a>	External Debug Peripheral Identification Register 4
0xFE0	<a href="#">EDPIDR0</a>	External Debug Peripheral Identification Register 0
0xFE4	<a href="#">EDPIDR1</a>	External Debug Peripheral Identification Register 1
0xFE8	<a href="#">EDPIDR2</a>	External Debug Peripheral Identification Register 2

Offset	Name	Description
0xFEC	<a href="#">EDPIDR3</a>	External Debug Peripheral Identification Register 3
0xFF0	<a href="#">EDCIDR0</a>	External Debug Component Identification Register 0
0xFF4	<a href="#">EDCIDR1</a>	External Debug Component Identification Register 1
0xFF8	<a href="#">EDCIDR2</a>	External Debug Component Identification Register 2
0xFFC	<a href="#">EDCIDR3</a>	External Debug Component Identification Register 3

## In the GIC Redistributor block:

Frame	Offset	Name	Description
RD_base	0x0000	<a href="#">GICR_CTLR</a>	Redistributor Control Register
RD_base	0x0004	<a href="#">GICR_IIDR</a>	Redistributor Implementer Identification Register
RD_base	0x0008–0x000C	<a href="#">GICR_TYPER</a>	Redistributor Type Register
RD_base	0x0010	<a href="#">GICR_STATUSR</a>	Error Reporting Status Register
RD_base	0x0014	<a href="#">GICR_WAKER</a>	Redistributor Wake Register
RD_base	0x0040–0x0044	<a href="#">GICR_SETLPIR</a>	Set LPI Pending Register
RD_base	0x0048–0x004C	<a href="#">GICR_CLRLPIR</a>	Clear LPI Pending Register
RD_base	0x0070–0x0074	<a href="#">GICR_PROPBASER</a>	Redistributor Properties Base Address Register
RD_base	0x0078–0x007C	<a href="#">GICR_PENDBASER</a>	Redistributor LPI Pending Table Base Address Register
RD_base	0x00A0–0x00A4	<a href="#">GICR_INVLPIR</a>	Redistributor Invalidate LPI Register
RD_base	0x00B0–0x00B4	<a href="#">GICR_INVALLR</a>	Redistributor Invalidate All Register
RD_base	0x00C0–0x00C4	<a href="#">GICR_SYNCR</a>	Redistributor Synchronize Register
VLPI_base	0x0070–0x0074	<a href="#">GICR_VPROPBASER</a>	Virtual Redistributor Properties Base Address Register
VLPI_base	0x0078–0x007C	<a href="#">GICR_VPENDBASER</a>	Virtual Redistributor LPI Pending Table Base Address Register
SGI_base	0x0080	<a href="#">GICR_IGROUPR0</a>	Interrupt Group Register 0
SGI_base	0x0100	<a href="#">GICR_ISENBALER0</a>	Interrupt Set-Enable Register 0
SGI_base	0x0180	<a href="#">GICR_ICENABLER0</a>	Interrupt Clear-Enable Register 0
SGI_base	0x0200	<a href="#">GICR_ISPENDR0</a>	Interrupt Set-Pending Register 0
SGI_base	0x0280	<a href="#">GICR_ICPENDR0</a>	Interrupt Clear-Pending Register 0
SGI_base	0x0300	<a href="#">GICR_ISACTIVER0</a>	Interrupt Set-Active Register 0
SGI_base	0x0380	<a href="#">GICR_ICACTIVER0</a>	Interrupt Clear-Active Register 0
SGI_base	0x0400 + 4n	<a href="#">GICR_IPRIORITYR&lt;n&gt;</a>	Interrupt Priority Registers
SGI_base	0x0C00	<a href="#">GICR_ICFGR0</a>	Interrupt Configuration Register 0
SGI_base	0x0C04	<a href="#">GICR_ICFGR1</a>	Interrupt Configuration Register 1
SGI_base	0x0D00	<a href="#">GICR_IGRPMODR0</a>	Interrupt Group Modifier Register 0
SGI_base	0x0E00	<a href="#">GICR_NSACR</a>	Non-secure Access Control Register

## In the GIC Virtual CPU interface block:

Offset	Name	Description
0x0000	<a href="#">GICV_CTLR</a>	Virtual Machine Control Register
0x0004	<a href="#">GICV_PMR</a>	Virtual Machine Priority Mask Register
0x0008	<a href="#">GICV_BPR</a>	Virtual Machine Binary Point Register
0x000C	<a href="#">GICV_IAR</a>	Virtual Machine Interrupt Acknowledge Register
0x0010	<a href="#">GICV_EOIR</a>	Virtual Machine End Of Interrupt Register
0x0014	<a href="#">GICV_RPR</a>	Virtual Machine Running Priority Register
0x0018	<a href="#">GICV_HPPIR</a>	Virtual Machine Highest Priority Pending Interrupt Register
0x001C	<a href="#">GICV_ABPR</a>	Virtual Machine Aliased Binary Point Register
0x0020	<a href="#">GICV_AIAR</a>	Virtual Machine Aliased Interrupt Acknowledge Register
0x0024	<a href="#">GICV_AEOIR</a>	Virtual Machine Aliased End Of Interrupt Register

Offset	Name	Description
0x0028	<a href="#">GICV_AHPPIR</a>	Virtual Machine Aliased Highest Priority Pending Interrupt Register
0x002C	<a href="#">GICV_STATUSR</a>	Virtual Machine Error Reporting Status Register
0x00D0 + 4n	<a href="#">GICV_APR&lt;n&gt;</a>	Virtual Machine Active Priorities Registers
0x00FC	<a href="#">GICV_IIDR</a>	Virtual Machine CPU Interface Identification Register
0x1000	<a href="#">GICV_DIR</a>	Virtual Machine Deactivate Interrupt Register

## In the GIC ITS control block:

Offset	Name	Description
0x0000	<a href="#">GITS_CTLR</a>	ITS Control Register
0x0004	<a href="#">GITS_IIDR</a>	ITS Identification Register
0x0008–0x000C	<a href="#">GITS_TYPER</a>	ITS Type Register
0x0080–0x0084	<a href="#">GITS_CBASER</a>	ITS Command Queue Descriptor
0x0088–0x008C	<a href="#">GITS_CWRITER</a>	ITS Write Register
0x0090–0x0094	<a href="#">GITS_CREADR</a>	ITS Read Register
0x0100 + 8n	<a href="#">GITS_BASER&lt;n&gt;</a>	ITS Translation Table Descriptors

## In the GIC ITS translation block:

Offset	Name	Description
0x0040	<a href="#">GITS_TRANSLATER</a>	ITS Translation Register

## In the CTI block:

Offset	Name	Description
0x000	<a href="#">CTICONTROL</a>	CTI Control register
0x010	<a href="#">CTIINTACK</a>	CTI Output Trigger Acknowledge register
0x014	<a href="#">CTIAPPSET</a>	CTI Application Trigger Set register
0x018	<a href="#">CTIAPPCLEAR</a>	CTI Application Trigger Clear register
0x01C	<a href="#">CTIAPPULSE</a>	CTI Application Pulse register
0x020 + 4n	<a href="#">CTIINEN&lt;n&gt;</a>	CTI Input Trigger to Output Channel Enable registers
0x0A0 + 4n	<a href="#">CTIOUTEN&lt;n&gt;</a>	CTI Input Channel to Output Trigger Enable registers
0x130	<a href="#">CTITRIGINSTATUS</a>	CTI Trigger In Status register
0x134	<a href="#">CTITRIGOUTSTATUS</a>	CTI Trigger Out Status register
0x138	<a href="#">CTICHINSTATUS</a>	CTI Channel In Status register
0x13C	<a href="#">CTICHOUTSTATUS</a>	CTI Channel Out Status register
0x140	<a href="#">CTIGATE</a>	CTI Channel Gate Enable register
0x144	<a href="#">ASICCTL</a>	CTI External Multiplexer Control register
0xF00	<a href="#">CTIITCTRL</a>	CTI Integration mode Control register
0xFA0	<a href="#">CTICLAIMSET</a>	CTI Claim Tag Set register
0xFA4	<a href="#">CTICLAIMCLR</a>	CTI Claim Tag Clear register
0xFA8	<a href="#">CTIDEVAFF0</a>	CTI Device Affinity register 0
0xFAC	<a href="#">CTIDEVAFF1</a>	CTI Device Affinity register 1
0xFB0	<a href="#">CTILAR</a>	CTI Lock Access Register
0xFB4	<a href="#">CTILSR</a>	CTI Lock Status Register
0xFB8	<a href="#">CTIAUTHSTATUS</a>	CTI Authentication Status register
0xFBC	<a href="#">CTIDEVARCH</a>	CTI Device Architecture register
0xFC0	<a href="#">CTIDEVID2</a>	CTI Device ID register 2
0xFC4	<a href="#">CTIDEVID1</a>	CTI Device ID register 1



Offset	Name	Description
0xFC8	<a href="#">CTIDEVID</a>	CTI Device ID register 0
0xFCC	<a href="#">CTIDEVTYPE</a>	CTI Device Type register
0xFD0	<a href="#">CTIPIDR4</a>	CTI Peripheral Identification Register 4
0xFE0	<a href="#">CTIPIDR0</a>	CTI Peripheral Identification Register 0
0xFE4	<a href="#">CTIPIDR1</a>	CTI Peripheral Identification Register 1
0xFE8	<a href="#">CTIPIDR2</a>	CTI Peripheral Identification Register 2
0xFEC	<a href="#">CTIPIDR3</a>	CTI Peripheral Identification Register 3
0xFF0	<a href="#">CTICIDR0</a>	CTI Component Identification Register 0
0xFF4	<a href="#">CTICIDR1</a>	CTI Component Identification Register 1
0xFF8	<a href="#">CTICIDR2</a>	CTI Component Identification Register 2
0xFFC	<a href="#">CTICIDR3</a>	CTI Component Identification Register 3

## In the GIC Distributor block:

Offset	Name	Description
0x0000	<a href="#">GICD_CTLR</a>	Distributor Control Register
0x0004	<a href="#">GICD_TYPER</a>	Interrupt Controller Type Register
0x0008	<a href="#">GICD_IIDR</a>	Distributor Implementer Identification Register
0x0010	<a href="#">GICD_STATUSR</a>	Error Reporting Status Register
0x0040	<a href="#">GICD_SETSPI_NSR</a>	Set Non-secure SPI Pending Register
0x0048	<a href="#">GICD_CLRSPI_NSR</a>	Clear Non-secure SPI Pending Register
0x0050	<a href="#">GICD_SETSPI_SR</a>	Set Secure SPI Pending Register
0x0058	<a href="#">GICD_CLRSPI_SR</a>	Clear Secure SPI Pending Register
0x0080 + 4n	<a href="#">GICD_IGROUPR&lt;n&gt;</a>	Interrupt Group Registers
0x0100 + 4n	<a href="#">GICD_ISENBALER&lt;n&gt;</a>	Interrupt Set-Enable Registers
0x0180 + 4n	<a href="#">GICD_ICENABLER&lt;n&gt;</a>	Interrupt Clear-Enable Registers
0x0200 + 4n	<a href="#">GICD_ISPENDR&lt;n&gt;</a>	Interrupt Set-Pending Registers
0x0280 + 4n	<a href="#">GICD_ICPENDR&lt;n&gt;</a>	Interrupt Clear-Pending Registers
0x0300 + 4n	<a href="#">GICD_ISACTIVER&lt;n&gt;</a>	Interrupt Set-Active Registers
0x0380 + 4n	<a href="#">GICD_ICACTIVER&lt;n&gt;</a>	Interrupt Clear-Active Registers
0x0400 + 4n	<a href="#">GICD_IPRIORITYR&lt;n&gt;</a>	Interrupt Priority Registers
0x0800 + 4n	<a href="#">GICD_ITARGETSR&lt;n&gt;</a>	Interrupt Processor Targets Registers
0x0C00 + 4n	<a href="#">GICD_ICFGR&lt;n&gt;</a>	Interrupt Configuration Registers
0x0D00 + 4n	<a href="#">GICD_IGRPMODR&lt;n&gt;</a>	Interrupt Group Modifier Registers
0x0E00 + 4n	<a href="#">GICD_NSACR&lt;n&gt;</a>	Non-secure Access Control Registers
0x0F00	<a href="#">GICD_SGIR</a>	Software Generated Interrupt Register
0x0F10 + 4n	<a href="#">GICD_CPENDSGIR&lt;n&gt;</a>	SGI Clear-Pending Registers
0x0F20 + 4n	<a href="#">GICD_SPENDSGIR&lt;n&gt;</a>	SGI Set-Pending Registers
0x6000 + 8n	<a href="#">GICD_IROUTER&lt;n&gt;</a>	Interrupt Routing Registers

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# ASICCTL, CTI External Multiplexer Control register

The ASICCTL characteristics are:

## Purpose

Can be used to provide IMPLEMENTATION DEFINED controls for the CTI. For example, the register might be used to control multiplexors for additional IMPLEMENTATION DEFINED triggers. The IMPLEMENTATION DEFINED controls provided by this register might modify the architecturally defined behavior of the CTI.

**Note**

The architecturally-defined triggers must not be multiplexed.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
IMP DEF	IMP DEF	IMP DEF	IMP DEF	RO	IMP DEF

## Configuration

It is IMPLEMENTATION DEFINED whether ASICCTL is implemented in the Core power domain or in the Debug power domain.

If it is implemented in the Core power domain then it is IMPLEMENTATION DEFINED whether it is in the Cold reset domain or the Warm reset domain.

This register must reset to a value that supports the architecturally-defined behavior of the CTI. Changing the value of the register from its reset value causes IMPLEMENTATION DEFINED behavior that might differ from the architecturally-defined behavior of the CTI.

Other than the requirements listed in this register description, all aspects of the reset behavior of the ASICCTL are IMPLEMENTATION DEFINED.

## Attributes

ASICCTL is a 32-bit register.

## Field descriptions

The ASICCTL 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
IMPLEMENTATION DEFINED																															

**IMPLEMENTATION DEFINED, bits [31:0]**

IMPLEMENTATION DEFINED.

## Accessing the ASICCTL

ASICCTL can be accessed through the external debug interface:



Component	Offset
CTI	0x144

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTACR<n>, Counter-timer Access Control Registers, n = 0 - 7

The CNTACR<n> characteristics are:

## Purpose

Provides top-level access controls for the elements of a timer frame. CNTACR<n> provides the controls for frame CNTBaseN.

In addition to the CNTACR<n> control:

- [CNTNSAR](#) controls whether CNTACR<n> is accessible by Non-secure accesses.
- If frame CNTEL0BaseN is implemented, the [CNTEL0ACR](#) in frame CNTBaseN provides additional control of accesses to frame CNTEL0BaseN.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RW

In a system that recognizes two Security states:

- CNTACR<n> is always accessible by Secure accesses.
- [CNTNSAR](#).NS<n> determines whether CNTACR<n> is accessible by Non-secure accesses.

## Configuration

The power domain of CNTACR<n> is IMPLEMENTATION DEFINED.

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.

Implemented only if the value of [CNTTIDR](#).Frame<n> is 1.

An implementation of the counters might not provide configurable access to some or all of the features. In this case, the associated field in the CNTACR<n> register is:

- RAZ/WI if access is always denied.
- RAO/WI if access is always permitted.

## Attributes

CNTACR<n> is a 32-bit register.

## Field descriptions

The CNTACR<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	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<a href="#">RWPT</a>	<a href="#">RWVT</a>	<a href="#">RVOFF</a>	<a href="#">RFRQ</a>	<a href="#">RVCT</a>	<a href="#">RPCT</a>

### Bits [31:6]

Reserved, RES0.

**RWPT, bit [5]**

Read/write access to the EL1 Physical Timer registers [CNTP\\_CVAL](#), [CNTP\\_TVAL](#), and [CNTP\\_CTL](#), in frame <n>. The possible values of this bit are:

RWPT	Meaning
0	No access to the EL1 Physical Timer registers in frame <n>. The registers are RES0.
1	Read/write access to the EL1 Physical Timer registers in frame <n>.

**RWVT, bit [4]**

Read/write access to the Virtual Timer register [CNTV\\_CVAL](#), [CNTV\\_TVAL](#), and [CNTV\\_CTL](#), in frame <n>. The possible values of this bit are:

RWVT	Meaning
0	No access to the Virtual Timer registers in frame <n>. The registers are RES0.
1	Read/write access to the Virtual Timer registers in frame <n>.

**RVOFF, bit [3]**

Read-only access to [CNTVOFF](#), in frame <n>. The possible values of this bit are:

RVOFF	Meaning
0	No access to <a href="#">CNTVOFF</a> in frame <n>. The register is RES0.
1	Read-only access to <a href="#">CNTVOFF</a> in frame <n>.

**RFRQ, bit [2]**

Read-only access to [CNTFRQ](#), in frame <n>. The possible values of this bit are:

RFRQ	Meaning
0	No access to <a href="#">CNTFRQ</a> in frame <n>. The register is RES0.
1	Read-only access to <a href="#">CNTFRQ</a> in frame <n>.

**RVCT, bit [1]**

Read-only access to [CNTVCT](#), in frame <n>. The possible values of this bit are:

RVCT	Meaning
0	No access to <a href="#">CNTVCT</a> in frame <n>. The register is RES0.
1	Read-only access to <a href="#">CNTVCT</a> in frame <n>.

**RPCT, bit [0]**

Read-only access to [CNTPCT](#), in frame <n>. The possible values of this bit are:

RPCT	Meaning
0	No access to <a href="#">CNTPCT</a> in frame <n>. The register is RES0.
1	Read-only access to <a href="#">CNTPCT</a> in frame <n>.

**Accessing the CNTACR<n>**

CNTACR<n> can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTCTLBase	0x040 + 4n

# CNTCR, Counter Control Register

The CNTCR characteristics are:

## Purpose

Enables the counter, controls the counter frequency setting, and controls counter behavior during debug.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RW

In a system that supports Secure and Non-secure memory maps the CNTControlBase frame, that includes this register, is implemented only in the Secure memory map.

## Configuration

The power domain of CNTCR is IMPLEMENTATION DEFINED.

Some or all fields in this register have defined reset values. These apply only on a reset of the reset domain in which the register is implemented. 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

CNTCR is a 32-bit register.

## Field descriptions

The CNTCR 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	FCREQ										0	0	0	0	0	0	HDBGEN		

### Bits [31:18]

Reserved, RES0.

### FCREQ, bits [17:8]

Frequency change request. Indicates the number of the entry in the Frequency modes table to select.

Selecting an unimplemented entry, or an entry that contains 0, has no effect on the counter.

The maximum number of entries in the Frequency modes table is IMPLEMENTATION DEFINED up to a maximum of 1004 entries, see 'The Frequency modes table' in Chapter I1 of the ARMv8 ARM. An implementation is only required to implement an FCREQ field that can hold values from 0 to the highest supported Frequency modes table entry. Any unrequired most-significant bits of FCREQ can be implemented as RES0.

When this register has an architecturally-defined reset value, this field resets to 0.

**Bits [7:2]**

Reserved, RES0.

**HDBG, bit [1]**

Halt-on-debug. Controls whether a Halt-on-debug signal halts the system counter:

HDBG	Meaning
0	System counter ignores Halt-on-debug.
1	Asserted Halt-on-debug signal halts system counter update.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**EN, bit [0]**

Enables the counter:

EN	Meaning
0	System counter disabled.
1	System counter enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

**Accessing the CNTCR**

CNTCR can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTControlBase	0x000

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTCV, Counter Count Value register

The CNTCV characteristics are:

## Purpose

Indicates the current count value.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default	
RW in CNTControlBase, RO in CNTReadBase	

Frame	Accessibility
CNTControlBase	RW
CNTReadBase	RO

A write to CNTCV must be visible in the [CNTPCT](#) register of each running processor in a finite time.

For the instance of the register in the CNTControlBase frame:

- In a system that supports Secure and Non-secure memory maps the CNTControlBase frame, and therefore this register instance, is implemented only in the Secure memory map.
- If the counter is enabled, the effect of writing to the register is UNKNOWN.

In an implementation that supports 64-bit atomic memory accesses, this register must be accessible using a 64-bit atomic access.

## Configuration

The power domain of CNTCV 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

CNTCV is a 64-bit register.

## Field descriptions

The CNTCV 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
																CountValue															
																CountValue															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### CountValue, bits [63:0]

Indicates the counter value.

## Accessing the CNTCV

CNTCV[31:0] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTControlBase	0x008
Timer	CNTReadBase	0x000

CNTCV[63:32] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTControlBase	0x00C
Timer	CNTReadBase	0x004

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTEL0ACR, Counter-timer EL0 Access Control Register

The CNTEL0ACR characteristics are:

## Purpose

An implementation of CNTEL0ACR in the frame at CNTBaseN controls whether the [CNTPCT](#), [CNTVCT](#), [CNTRQ](#), EL1 Physical Timer, and Virtual Timer registers are visible in the frame at CNTEL0BaseN.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RW

CNTEL0ACR can be implemented in any implemented CNTBaseN 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.

If CNTEL0ACR is not implemented in an implemented CNTBaseN frame:

- The register location in that frame is RAZ/WI.
- If the corresponding CNTEL0BaseN frame is implemented, the registers [CNTRQ](#), [CNTP\\_CTL](#), [CNTP\\_CVAL](#), [CNTP\\_TVAL](#), [CNTPCT](#), [CNTV\\_CTL](#), [CNTV\\_CVAL](#), [CNTV\\_TVAL](#), and [CNTVCT](#) are not visible in that frame.

## Configuration

The power domain of CNTEL0ACR is IMPLEMENTATION DEFINED.

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.

Implementation of this register is OPTIONAL.

## Attributes

CNTEL0ACR is a 32-bit register.

## Field descriptions

The CNTEL0ACR 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	ELOPTEN	ELOVTEN	0	0	0	0	0	0	ELOVCTEN	ELOPCTEN

Bits [31:10]

Reserved, RES0.



**EL0PTEN, bit [9]**

Second view read/write access control for the EL1 Physical Timer registers. This bit controls whether the [CNTP\\_CVAL](#), [CNTP\\_TVAL](#), and [CNTP\\_CTL](#) registers in the current CNTBaseN frame are also accessible in the corresponding CNTEL0BaseN frame. The possible values of this bit are:

EL0PTEN	Meaning
0	No access. Registers are RES0 in the second view.
1	Access permitted. If the registers are accessible in the current frame then they are accessible in the second view.

**EL0VTEN, bit [8]**

Second view read/write access control for the Virtual Timer registers. This bit controls whether the [CNTV\\_CVAL](#), [CNTV\\_TVAL](#), and [CNTV\\_CTL](#) registers in the current CNTBaseN frame are also accessible in the corresponding CNTEL0BaseN frame. The possible values of this bit are:

EL0VTEN	Meaning
0	No access. Registers are RES0 in the second view.
1	Access permitted. If the registers are accessible in the current frame then they are accessible in the second view.

The definition of this bit means that, if the Virtual Timer registers are not implemented in the current CNTBaseN frame, then the Virtual Timer register addresses are RES0 in the corresponding CNTEL0BaseN frame, regardless of the value of this bit.

**Bits [7:2]**

Reserved, RES0.

**EL0VCTEN, bit [1]**

Second view read access control for [CNTVCT](#) and [CNTFRQ](#). The possible values of this bit are:

EL0VCTEN	Meaning
0	<a href="#">CNTVCT</a> is not visible in the second view. If EL0PTEN is set to 0, <a href="#">CNTFRQ</a> is not visible in the second view.
1	Access permitted. If <a href="#">CNTVCT</a> and <a href="#">CNTFRQ</a> are visible in the current frame then they are visible in the second view.

**EL0PCTEN, bit [0]**

Second view read access control for [CNTPCT](#) and [CNTFRQ](#). The possible values of this bit are:

EL0PCTEN	Meaning
0	<a href="#">CNTPCT</a> is not visible in the second view. If EL0VCTEN is set to 0, <a href="#">CNTFRQ</a> is not visible in the second view.
1	Access permitted. If <a href="#">CNTPCT</a> and <a href="#">CNTFRQ</a> are visible in the current frame then they are visible in the second view.

**Accessing the CNTEL0ACR**

CNTEL0ACR can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x014

# CNTFID0, Counter Frequency ID

The CNTFID0 characteristics are:

## Purpose

Indicates the base frequency of the system counter.

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 this register, is implemented only in the Secure memory map.

## Configuration

The power domain of CNTFID0 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. For more information see 'The Frequency modes table' in Chapter I1 of the ARMv8 ARM.

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

CNTFID0 is a 32-bit register.

## Field descriptions

The CNTFID0 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]

The base frequency of the system counter, in Hz.

## Accessing the CNTFID0

CNTFID0 can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTControlBase	0x020

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTFID<n>, Counter Frequency IDs

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:

increment = (base frequency) / (selected frequency)

## Accessing the CNTFID<n>

CNTFID<n> can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTControlBase	$0 \times 020 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTFRQ, Counter-timer Frequency

The CNTFRQ characteristics are:

## Purpose

This register is provided so that software can discover the frequency of the system counter. The instance of the register in the CNTCTLBase frame must be programmed with this value as part of system initialization. The value of the register is not interpreted by hardware.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

CNTFRQ must be implemented as an RW register in the CNTCTLBase frame.

In a system that recognizes two Security states, the instance of the register in the CNTCTLBase frame is only accessible by Secure accesses.

CNTFRQ can be implemented as a RO register 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:

- CNTFRQ is accessible in that frame, as a RO register, if the value of [CNTACR<n>.RFRQ](#) is 1.
- Otherwise, the CNTFRQ address in that frame is RAZ/WI.

For an implemented CNTEL0BaseN frame:

- CNTFRQ is accessible as a RO register in that frame if both:
  - CNTFRQ is accessible in the corresponding CNTBaseN frame.
  - Either the value of [CNTEL0ACR.EL0VC TEN](#) is 1 or the value of [CNTEL0ACR.EL0PCTEN](#) is 1.
- Otherwise, the CNTFRQ address in that frame is RAZ/WI.

## Configuration

The power domain of CNTFRQ is IMPLEMENTATION DEFINED.

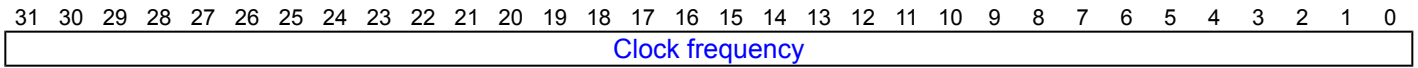
On a reset of the reset domain in which an RW instance of this register is implemented, RW fields in the register reset to UNKNOWN values. The register is not affected by a reset of any other reset domain. For more information see 'Power and reset domains for the system level implementation of the Generic Timer' in Chapter I1 of the ARMv8 ARM.

## Attributes

CNTFRQ is a 32-bit register.

## Field descriptions

The CNTFRQ bit assignments are:

**Bits [31:0]**

Clock frequency. Indicates the system counter clock frequency, in Hz.

## Accessing the CNTFRQ

CNTFRQ can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x010
Timer	CNTELOBaseN	0x010
Timer	CNTCTLBase	0x000

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTNSAR, Counter-timer Non-secure Access Register

The CNTNSAR characteristics are:

## Purpose

Provides the highest-level control of whether frames CNTBaseN and CNTELOBaseN are accessible by Non-secure accesses.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RW

In a system that recognizes two Security states, this register is only accessible by Secure accesses.

## Configuration

The power domain of CNTNSAR is IMPLEMENTATION DEFINED.

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.

## Attributes

CNTNSAR is a 32-bit register.

## Field descriptions

The CNTNSAR 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	NS7	NS6	NS5	NS4	NS3	NS2	NS1	NS0

### Bits [31:8]

Reserved, RES0.

### NS<n>, bit [n], for n = 0 to 7

Non-secure access to frame n. The possible values of this bit are:

NS<n>	Meaning
0	Secure access only. Behaves as RES0 to Non-secure accesses.
1	Secure and Non-secure accesses permitted.

This bit also determines whether, in the CNTCTLBase frame, [CNTACR<n>](#) and [CNTVOFF<n>](#) are accessible to Non-secure accesses.

If frame CNTBase<n>:

- Is not implemented, then NS<n> is RES0.
- Is not Configurable access, and is accessible only by Secure accesses, then NS<n> is RES0.
- Is not Configurable access, and is accessible by both Secure and Non-secure accesses, then NS<n> is RES1.



## Accessing the CNTNSAR

CNTNSAR can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTCTLBase	0x004

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTPCT, Counter-timer Physical Count

The CNTPCT characteristics are:

## Purpose

Holds the 64-bit physical count value.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

CNTPCT can be implemented in any implemented CNTBaseN frame, and in the corresponding CNTEL0BaseN frame, as a RO register.

'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:

- CNTPCT is accessible in that frame, as a RO register, if the value of [CNTACR<n>.RPCT](#) is 1.
- Otherwise, the CNTPCT address in that frame is RAZ/WI.

For an implemented CNTEL0BaseN frame:

- CNTPCT is accessible in that frame if both:
  - CNTPCT is accessible in the corresponding CNTBaseN frame.
  - The value of [CNTEL0ACR.EL0PCTEN](#) is 1.
- Otherwise, the CNTPCT address in that frame is RAZ/WI.

If the implementation supports 64-bit atomic accesses, then the CNTPCT register must be accessible as an atomic 64-bit value.

## Configuration

The power domain of CNTPCT is IMPLEMENTATION DEFINED.

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

CNTPCT is a 64-bit register.

## Field descriptions

The CNTPCT 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
Physical count value																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**Bits [63:0]**

Physical count value.

## Accessing the CNTPCT

CNTPCT[31:0] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x000
Timer	CNTELOBaseN	0x000

CNTPCT[63:32] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x004
Timer	CNTELOBaseN	0x004

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTP\_CTL, Counter-timer Physical Timer Control

The CNTP\_CTL characteristics are:

## Purpose

Control register 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\_CTL 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\_CTL is accessible in that frame if the value of [CNTACR<n>](#).RWPT is 1.
- Otherwise, the CNTP\_CTL address in that frame is RAZ/WI.

For an implemented CNTEL0BaseN frame:

- CNTP\_CTL is accessible in that frame if both:
  - CNTP\_CTL is accessible in the corresponding CNTBaseN frame:
  - The value of [CNTEL0ACR](#).ELOPTEN is 1.
- Otherwise, the CNTP\_CTL address in that frame is RAZ/WI.

## Configuration

The power domain of CNTP\_CTL 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\_CTL is a 32-bit register.

## Field descriptions

The CNTP\_CTL 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	0
																STATUS				MASK				ENABLE							

**Bits [31:3]**

Reserved, RES0.

**ISTATUS, bit [2]**

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0	Timer condition is not met.
1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.

For more information see 'Operation of the CompareValue views of the timers' and 'Operation of the TimerValue views of the timers' in the ARM ARM, chapter D6.

This bit is read-only.

**IMASK, bit [1]**

Timer interrupt mask bit. Permitted values are:

IMASK	Meaning
0	Timer interrupt is not masked by the IMASK bit.
1	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

**ENABLE, bit [0]**

Enables the timer. Permitted values are:

ENABLE	Meaning
0	Timer disabled.
1	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from [CNTP\\_TVAL](#) continues to count down.

**Note**

Disabling the output signal might be a power-saving option.

**Accessing the CNTP\_CTL**

CNTP\_CTL can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x02C
Timer	CNTELOBaseN	0x02C

# 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 CNTELOBaseN frame.

'CNTCTLBase status and control fields for the CNTBaseN and CNTELOBaseN 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 CNTELOBaseN frame is implemented.
- For an implementation that recognizes two Security states, whether the CNTBaseN frame, and any corresponding CNTELOBaseN 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 CNTELOBaseN frame:

- CNTP\_CVAL is accessible in that frame if both:
  - CNTP\_CVAL is accessible in the corresponding CNTBaseN frame:
  - The value of [CNTELOACR.ELOPTEN](#) 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.

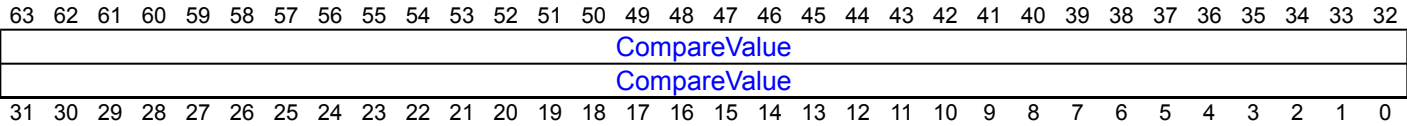
On a reset of the reset domain in which an RW instance of this register is implemented, RW fields in the register reset to UNKNOWN values. The register is not affected by a reset of any other reset domain. For more information see 'Power and reset domains for the system level implementation of the Generic 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 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

# 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 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	CNTELOBaseN	0x028

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTSR, Counter Status Register

The CNTSR characteristics are:

## Purpose

Provides counter frequency status information.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

In a system that supports Secure and Non-secure memory maps the CNTControlBase frame, that includes this register, is implemented only in the Secure memory map.

## Configuration

The power domain of CNTSR is IMPLEMENTATION DEFINED.

Some or all fields in this register have defined reset values. These apply only on a reset of the reset domain in which the register is implemented. 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

CNTSR is a 32-bit register.

## Field descriptions

The CNTSR 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
								FCACK																0	0	0	0	0	0	DBGH	0

### FCACK, bits [31:8]

Frequency change acknowledge. Indicates the currently selected entry in the Frequency modes table, see 'The Frequency modes table' in Chapter I1 of the ARMv8 ARM.

When this register has an architecturally-defined reset value, this field resets to 0.

### Bits [7:2]

Reserved, RES0.

### DBGH, bit [1]

Indicates whether the counter is halted because the Halt-on-Debug signal is asserted:

DBGH	Meaning
0	Counter is not halted.
1	Counter is halted.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Bit [0]

Reserved, RES0.

## Accessing the CNTSR

CNTSR can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTControlBase	0x004

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTTIDR, Counter-timer Timer ID Register

The CNTTIDR characteristics are:

## Purpose

Indicates the implemented timers in the memory map, and their features. For each value of N from 0 to 7 it indicates whether:

- Frame CNTBaseN is a view of an implemented timer.
- Frame CNTBaseN has a second view, CNTEL0BaseN.
- Frame CNTBaseN has a virtual timer capability.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

In a system that recognizes two Security states this register is accessible by both Secure and Non-secure accesses.

## Configuration

The power domain of CNTTIDR is IMPLEMENTATION DEFINED.

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

CNTTIDR is a 32-bit register.

## Field descriptions

The CNTTIDR 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
Frame7	Frame6	Frame5	Frame4	Frame3	Frame2	Frame1	Frame0																								

**Frame<n>, bits [4n+3:4n], for n = 0 to 7**

A 4-bit field indicating the features of frame CNTBase<n>.

Bit[3] of the field is RES0.

Bit[2] indicates whether frame CNTBase<n> has a second view, CNTEL0Base<n>. The possible values of this bit are:

Bit[2]	Meaning
0	Frame <n> does not have a second view. CNTEL0Base<n> is RES0.
1	Frame <n> has a second view, CNTEL0Base<n>.

If bit[0] is 0, bit[2] is RES0.

Bit[1] indicates whether both:

- Frame CNTBase<n> implements the virtual timer registers [CNTV\\_CVAL](#), [CNTV\\_TVAL](#), and [CNTV\\_CTL](#).
- This CNTCTLBase frame implements the virtual timer offset register [CNTVOFF<n>](#).

The possible values of bit[1] are:

Bit[1]	Meaning
0	Frame <n> does not have virtual capability. The virtual time and offset registers are RES0.
1	Frame <n> has virtual capability. The virtual time and offset registers are implemented.

If bit[0] is 0, bit[1] is RES0.

Bit[0] indicates whether frame CNTBase<n> is implemented. The possible values of this bit are:

Bit[0]	Meaning
0	Frame <n> not implemented. All registers associated with the frame are RES0.
1	Frame <n> is implemented.

## Accessing the CNTTIDR

CNTTIDR can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTCTLBase	0x008

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTVCT, Counter-timer Virtual Count

The CNTVCT characteristics are:

## Purpose

Holds the 64-bit virtual count value.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

CNTVCT can be implemented in any implemented CNTBaseN frame, and in the corresponding CNTEL0BaseN frame, as a RO register.

'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:

- CNTVCT is accessible in that frame, as a RO register, if the value of [CNTACR<n>.RVCT](#) is 1.
- Otherwise, the CNTVCT address in that frame is RAZ/WI.

For an implemented CNTEL0BaseN frame:

- CNTVCT is accessible in that frame if both:
  - CNTVCT is accessible in the corresponding CNTBaseN frame.
  - The value of [CNTEL0ACR.EL0VCTEN](#) is 1.
- Otherwise, the CNTVCT address in that frame is RAZ/WI.

If the implementation supports 64-bit atomic accesses, then the CNTVCT register must be accessible as an atomic 64-bit value.

## Configuration

The power domain of CNTVCT is IMPLEMENTATION DEFINED.

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

CNTVCT is a 64-bit register.

## Field descriptions

The CNTVCT 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
Virtual count value																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**Bits [63:0]**

Virtual count value.

## Accessing the CNTVCT

CNTVCT[31:0] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x008
Timer	CNTELOBaseN	0x008

CNTVCT[63:32] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x00C
Timer	CNTELOBaseN	0x00C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTVOFF, Counter-timer Virtual Offset

The CNTVOFF characteristics are:

## Purpose

Holds the 64-bit virtual offset for a CNTBaseN frame that has virtual timer capability. This is the offset between real time and virtual time.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

In the CNTCTLBase frame a CNTVOFF<n> register must be implemented, as a RW register, for each CNTBaseN frame that has virtual timer capability.

---

### Note

The value of <n> in an instance of CNTVOFF<n> specifies the value of N for the associated CNTBaseN frame.

In a system that recognizes two Security states, for any CNTVOFF<n> register in the CNTCTLBase frame:

- CNTVOFF<n> is always accessible by Secure accesses.
- [CNTNSAR.NS<n>](#) determines whether CNTVOFF<n> is accessible by Non-secure accesses.

The register location of any unimplemented CNTVOFF<n> register in the CNTCTLBase frame is RAZ/WI.

CNTVOFF is implemented, as a RO register, in any implemented CNTBaseN frame that has virtual timer capability.

'CNTCTLBase status and control fields for the CNTBaseN and CNTEL0BaseN frames' in Chapter 11 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:

- CNTVOFF is accessible in that frame, as a RO register, if the value of [CNTACR<n>.RVOFF](#) is 1.
- Otherwise, the CNTVOFF address in that frame is RAZ/WI.

---

### Note

CNTVOFF is never visible in any CNTEL0BaseN frame. This means that the CNTVOFF address in any implemented CNTEL0BaseN frame is RAZ/WI.

In an implementation that supports 64-bit atomic accesses, a CNTVOFF{<n>} register must be accessible as an atomic 64-bit value.

## Configuration

The power domain of CNTVOFF is IMPLEMENTATION DEFINED.



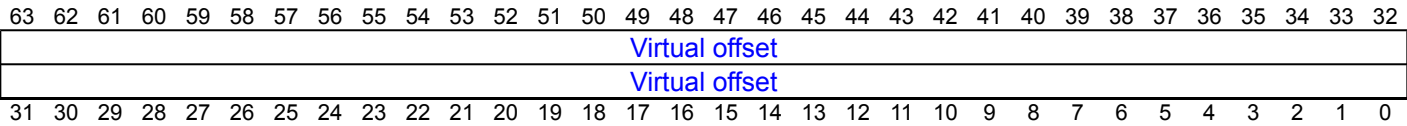
On a reset of the reset domain in which an RW instance of this register is implemented, RW fields in the register reset to UNKNOWN values. The register is not affected by a reset of any other reset domain. For more information see 'Power and reset domains for the system level implementation of the Generic Timer' in Chapter I1 of the ARMv8 ARM.

Attributes

CNTVOFF is a 64-bit register.

Field descriptions

The CNTVOFF bit assignments are:



Bits [63:0]

Virtual offset.

Accessing the CNTVOFF

CNTVOFF[31:0] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x018

CNTVOFF[63:32] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x01C

# CNTVOFF<n>, Counter-timer Virtual Offsets, n = 0 - 7

The CNTVOFF<n> characteristics are:

## Purpose

Holds the 64-bit virtual offset for frame CNTBase<n>.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RW

In the CNTCTLBase frame a CNTVOFF<n> register must be implemented, as a RW register, for each CNTBaseN frame that has virtual timer capability.

### Note

The value of <n> in an instance of CNTVOFF<n> specifies the value of N for the associated CNTBaseN frame.

In a system that recognizes two Security states, for any CNTVOFF<n> register in the CNTCTLBase frame:

- CNTVOFF<n> is always accessible by Secure accesses.
- [CNTNSAR.NS<n>](#) determines whether CNTVOFF<n> is accessible by Non-secure accesses.

The register location of any unimplemented CNTVOFF<n> register in the CNTCTLBase frame is RAZ/WI.

The CNTVOFF<n> register is accessible in the CNTBaseN frame using [CNTVOFF](#).

In an implementation that supports 64-bit atomic accesses, then the CNTVOFF<n> registers must be accessible as atomic 64-bit values.

## Configuration

The power domain of CNTVOFF<n> 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

CNTVOFF<n> is a 64-bit register.

## Field descriptions

The CNTVOFF<n> 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
																Virtual offset															
																Virtual offset															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

**Bits [63:0]**

Virtual offset.

**Accessing the CNTVOFF<n>**

CNTVOFF<n>[31:0] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTCTLBase	$0 \times 080 + 8n$

CNTVOFF<n>[63:32] can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTCTLBase	$0 \times 084 + 8n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CNTV\_CTL, Counter-timer Virtual Timer Control

The CNTV\_CTL characteristics are:

## Purpose

Control register 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\_CTL 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\_CTL is accessible in that frame if the value of [CNTACR<n>](#).RWVT is 1.
- Otherwise, the CNTV\_CTL address in that frame is RAZ/WI.

For an implemented CNTEL0BaseN frame:

- CNTV\_CTL is accessible in that frame if both:
  - CNTV\_CTL is accessible in the corresponding CNTBaseN frame:
  - The value of [CNTEL0ACR](#).EL0VTEN is 1.
- Otherwise, the CNTV\_CTL address in that frame is RAZ/WI.

## Configuration

The power domain of CNTV\_CTL 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\_CTL is a 32-bit register.

## Field descriptions

The CNTV\_CTL 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	ISTATUS	IMASK	ENABLE

**Bits [31:3]**

Reserved, RES0.

**ISTATUS, bit [2]**

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0	Timer condition is not met.
1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is UNKNOWN.

For more information see 'Operation of the CompareValue views of the timers' and 'Operation of the TimerValue views of the timers' in the ARM ARM, chapter D6.

This bit is read-only.

**IMASK, bit [1]**

Timer interrupt mask bit. Permitted values are:

IMASK	Meaning
0	Timer interrupt is not masked by the IMASK bit.
1	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

**ENABLE, bit [0]**

Enables the timer. Permitted values are:

ENABLE	Meaning
0	Timer disabled.
1	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from [CNTV\\_TVAL](#) continues to count down.

**Note**

Disabling the output signal might be a power-saving option.

**Accessing the CNTV\_CTL**

CNTV\_CTL can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTBaseN	0x03C
Timer	CNTELOBaseN	0x03C

# 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 CNTELOBaseN frame.

'CNTCTLBase status and control fields for the CNTBaseN and CNTELOBaseN 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 CNTELOBaseN frame is implemented.
- For an implementation that recognizes two Security states, whether the CNTBaseN frame, and any corresponding CNTELOBaseN 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 CNTELOBaseN frame:

- CNTV\_CVAL is accessible in that frame if both:
  - CNTV\_CVAL is accessible in the corresponding CNTBaseN frame:
  - The value of [CNTELOACR.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.

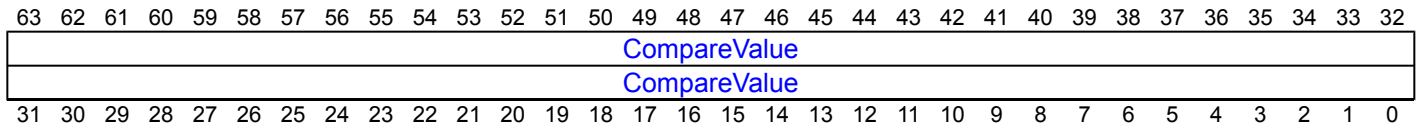
On a reset of the reset domain in which an RW instance of this register is implemented, RW fields in the register reset to UNKNOWN values. The register is not affected by a reset of any other reset domain. For more information see 'Power and reset domains for the system level implementation of the Generic 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 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

# 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 CNTELOBaseN frame.

'CNTCTLBase status and control fields for the CNTBaseN and CNTELOBaseN 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 CNTELOBaseN frame is implemented.
- For an implementation that recognizes two Security states, whether the CNTBaseN frame, and any corresponding CNTELOBaseN 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 CNTELOBaseN frame:

- CNTV\_TVAL is accessible in that frame if both:
  - CNTV\_TVAL is accessible in the corresponding CNTBaseN frame:
  - The value of [CNTELOACR.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:

31	30	29	28	27	26	25	24	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 (CompareValue - CNTVCT0).

On a write of this register, CompareValue is set to (CNTVCT0 + 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 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

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIAPPCLEAR, CTI Application Trigger Clear register

The CTIAPPCLEAR characteristics are:

## Purpose

Clears bits of the Application Trigger register.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	WO

## Configuration

CTIAPPCLEAR is in the Debug power domain.

## Attributes

CTIAPPCLEAR is a 32-bit register.

## Field descriptions

The CTIAPPCLEAR 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
APPCLEAR<x>, bit [x]																															

### APPCLEAR<x>, bit [x], for x = 0 to 31

Application trigger <x> disable.

Bits [31:N] are RAZ/WI. N is the number of ECT channels implemented as defined by the [CTIDEVID.NUMCHAN](#) field.

Writing to this bit has the following effect:

APPCLEAR<x>	Meaning
0	No effect.
1	Clear corresponding bit in CTIAPPTRIG to 0 and clear the corresponding channel event.

If the ECT does not support multicycle channel events, use of CTIAPPCLEAR is deprecated and the debugger must only use [CTIAPPPULSE](#).

## Accessing the CTIAPPCLEAR

CTIAPPCLEAR can be accessed through the external debug interface:

Component	Offset
CTI	0x018



# CTIAPPPULSE, CTI Application Pulse register

The CTIAPPPULSE characteristics are:

## Purpose

Causes event pulses to be generated on ECT channels.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	WO

It is CONSTRAINED UNPREDICTABLE whether a write to CTIAPPPULSE generates an event on a channel if CTICONTROL.GLBEN is 0.

## Configuration

CTIAPPPULSE is in the Debug power domain.

## Attributes

CTIAPPPULSE is a 32-bit register.

## Field descriptions

The CTIAPPPULSE 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
APPPULSE<x>, bit [x]																															

### APPPULSE<x>, bit [x], for x = 0 to 31

Generate event pulse on ECT channel <x>.

Bits [31:N] are RAZ/WI. N is the number of ECT channels implemented as defined by the [CTIDEVID.NUMCHAN](#) field.

Writing to this bit has the following effect:

APPPULSE<x>	Meaning
0	No effect.
1	Channel <x> event pulse generated.

#### Note

- The CTIAPPPULSE operation does not affect the state of the Application Trigger register, CTIAPPTRIG. If the channel is active, either because of an earlier event or from the application trigger, then the value written to CTIAPPPULSE might have no effect.
- Multiple pulse events that occur close together might be merged into a single pulse event.

## Accessing the CTIAPPPULSE

CTIAPPPULSE can be accessed through the external debug interface:

Component	Offset
CTI	0x01C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIAPPSET, CTI Application Trigger Set register

The CTIAPPSET characteristics are:

## Purpose

Sets bits of the Application Trigger register.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RW

## Configuration

CTIAPPSET is in the Debug power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply only on an External debug reset. The register is not affected by a Warm reset and is not affected by a Cold reset.

## Attributes

CTIAPPSET is a 32-bit register.

## Field descriptions

The CTIAPPSET 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
APPSET<x>, bit [x]																															

### APPSET<x>, bit [x], for x = 0 to 31

Application trigger <x> enable.

Bits [31:N] are RAZ/WI. N is the number of ECT channels implemented as defined by the [CTIDEVID.NUMCHAN](#) field.

Possible values of this bit are:

APPSET<x>	Meaning
0	Reading this means the application trigger is inactive. Writing this has no effect.
1	Reading this means the application trigger is active. Writing this sets the corresponding bit in CTIAPPTRIG to 1 and generates a channel event.

If the ECT does not support multicycle channel events, use of CTIAPPSET is deprecated and the debugger must only use [CTIAPPULSE](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the CTIAPPSET

CTIAPPSET can be accessed through the external debug interface:

Component	Offset
-----------	--------

CTI	0x014
-----	-------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIAUTHSTATUS, CTI Authentication Status register

The CTIAUTHSTATUS characteristics are:

## Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for CTI.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIAUTHSTATUS is in the Debug power domain.

This register is OPTIONAL, and is required for CoreSight compliance.

## Attributes

CTIAUTHSTATUS is a 32-bit register.

## Field descriptions

The CTIAUTHSTATUS 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	NSNID	NSID		

### Bits [31:8]

Reserved, RES0.

### Bits [7:4]

Reserved, RAZ.

### NSNID, bits [3:2]

If EL3 is not implemented and the implemented Security state is Secure state, holds the same value as [DBGAUTHSTATUS\\_EL1](#).SNID.

Otherwise, holds the same value as [DBGAUTHSTATUS\\_EL1](#).NSNID.

### NSID, bits [1:0]

If EL3 is not implemented and the implemented Security state is Secure state, holds the same value as [DBGAUTHSTATUS\\_EL1](#).SID.

Otherwise, holds the same value as [DBGAUTHSTATUS\\_EL1](#).NSID.



## Accessing the CTIAUTHSTATUS

CTIAUTHSTATUS can be accessed through the external debug interface:

Component	Offset
CTI	0xFB8

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTICHINSTATUS, CTI Channel In Status register

The CTICHINSTATUS characteristics are:

## Purpose

Provides the raw status of the ECT channel inputs to the CTI.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTICHINSTATUS is in the Debug power domain.

## Attributes

CTICHINSTATUS is a 32-bit register.

## Field descriptions

The CTICHINSTATUS 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
CHIN<n>, bit [n]																															

### CHIN<n>, bit [n], for n = 0 to 31

Input channel <n> status.

Bits [31:N] are RAZ. N is the number of ECT channels implemented as defined by the [CTIDEVID.NUMCHAN](#) field.

Possible values of this bit are:

CHIN<n>	Meaning
0	Input channel <n> is inactive.
1	Input channel <n> is active.

If the ECT channels do not support multicycle events then it is IMPLEMENTATION DEFINED whether an input channel can be observed as active.

## Accessing the CTICHINSTATUS

CTICHINSTATUS can be accessed through the external debug interface:

Component	Offset
CTI	0x138



# CTICHOUTSTATUS, CTI Channel Out Status register

The CTICHOUTSTATUS characteristics are:

## Purpose

Provides the status of the ECT channel outputs from the CTI.  
This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTICHOUTSTATUS is in the Debug power domain.

## Attributes

CTICHOUTSTATUS is a 32-bit register.

## Field descriptions

The CTICHOUTSTATUS 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
																CHOUT<n>, bit [n]															

### CHOUT<n>, bit [n], for n = 0 to 31

Output channel <n> status.  
Bits [31:N] are RAZ. N is the number of ECT channels implemented as defined by the [CTIDEVID](#).NUMCHAN field.  
Possible values of this bit are:

CHOUT<n>	Meaning
0	Output channel <n> is inactive.
1	Output channel <n> is active.

If the ECT channels do not support multicycle events then it is IMPLEMENTATION DEFINED whether an input channel can be observed as active.

### Note

The value in CTICHOUTSTATUS is after gating by the channel gate. For more information, see [CTIGATE](#).

## Accessing the CTICHOUTSTATUS

CTICHOUTSTATUS can be accessed through the external debug interface:

Component	Offset
CTI	0x13C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTICIDR0, CTI Component Identification Register 0

The CTICIDR0 characteristics are:

## Purpose

Provides information to identify a CTI component.

For more information see 'About the Component identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTICIDR0 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

CTICIDR0 is a 32-bit register.

## Field descriptions

The CTICIDR0 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	PRMBL_0									

### Bits [31:8]

Reserved, RES0.

### PRMBL\_0, bits [7:0]

Preamble. Must read as 0x0D.

## Accessing the CTICIDR0

CTICIDR0 can be accessed through the external debug interface:

Component	Offset
CTI	0xFF0



# CTICIDR1, CTI Component Identification Register 1

The CTICIDR1 characteristics are:

## Purpose

Provides information to identify a CTI component.

For more information see 'About the Component identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTICIDR1 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

CTICIDR1 is a 32-bit register.

## Field descriptions

The CTICIDR1 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	CLASS				PRMBL_1			

### Bits [31:8]

Reserved, RES0.

### CLASS, bits [7:4]

Component class. Reads as 0x9, debug component.

### PRMBL\_1, bits [3:0]

Preamble. RAZ.

## Accessing the CTICIDR1

CTICIDR1 can be accessed through the external debug interface:

Component	Offset
-----------	--------



CTI	0xFF4
-----	-------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTICIDR2, CTI Component Identification Register 2

The CTICIDR2 characteristics are:

## Purpose

Provides information to identify a CTI component.

For more information see 'About the Component identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTICIDR2 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

CTICIDR2 is a 32-bit register.

## Field descriptions

The CTICIDR2 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	PRMBL_2									

### Bits [31:8]

Reserved, RES0.

### PRMBL\_2, bits [7:0]

Preamble. Must read as 0x05.

## Accessing the CTICIDR2

CTICIDR2 can be accessed through the external debug interface:

Component	Offset
CTI	0xFF8



# CTICIDR3, CTI Component Identification Register 3

The CTICIDR3 characteristics are:

## Purpose

Provides information to identify a CTI component.

For more information see 'About the Component identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTICIDR3 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

CTICIDR3 is a 32-bit register.

## Field descriptions

The CTICIDR3 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	PRMBL_3							

### Bits [31:8]

Reserved, RES0.

### PRMBL\_3, bits [7:0]

Preamble. Must read as 0xB1.

## Accessing the CTICIDR3

CTICIDR3 can be accessed through the external debug interface:

Component	Offset
CTI	0xFFC



# CTICLAIMCLR, CTI Claim Tag Clear register

The CTICLAIMCLR characteristics are:

## Purpose

Used by software to read the values of the CLAIM bits, and to clear these bits to 0.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RW

## Configuration

CTICLAIMCLR is in the Debug power domain.

See the CLAIM field description for the effect of an External debug reset on the value returned by this register. This register is not affected by a Warm reset, and is not affected by a Cold reset.

Implementation of this register is OPTIONAL.

## Attributes

CTICLAIMCLR is a 32-bit register.

## Field descriptions

The CTICLAIMCLR 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
CLAIM[x], bit [x]																															

### CLAIM[x], bit [x], for = 0 to 31

CLAIM tag clear bit.

For values of x greater than or equal to the IMPLEMENTATION DEFINED number of CLAIM tags, this bit is RAZ/SBZ. Software can rely on these bits reading as zero, and must use a Should-Be-Zero policy on writes. Implementations must ignore writes.

For other values of x, reads return the value of CLAIM[x] and the behavior on writes is:

CLAIM[x]	Meaning
0	No action.
1	Indirectly clear CLAIM[x] to 0.

A single write to CTICLAIMCLR can clear multiple tags to 0.

An External Debug reset clears the CLAIM tag bits to 0.

## Accessing the CTICLAIMCLR

CTICLAIMCLR can be accessed through the external debug interface:

Component	Offset
CTI	0xFA4

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTICLAIMSET, CTI Claim Tag Set register

The CTICLAIMSET characteristics are:

## Purpose

Used by software to set CLAIM bits to 1.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RW

## Configuration

CTICLAIMSET is in the Debug power domain.

Implementation of this register is OPTIONAL.

## Attributes

CTICLAIMSET is a 32-bit register.

## Field descriptions

The CTICLAIMSET 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
CLAIM[x], bit [x]																															

### CLAIM[x], bit [x], for = 0 to 31

CLAIM tag set bit.

For values of x greater than or equal to the IMPLEMENTATION DEFINED number of CLAIM tags, this bit is RAZ/SBZ. Software can rely on these bits reading as zero, and must use a Should-Be-Zero policy on writes. Implementations must ignore writes.

For other values of x, the bit is RAO and the behavior on writes is:

CLAIM[x]	Meaning
0	No action.
1	Indirectly set CLAIM[x] tag to 1.

A single write to CTICLAIMSET can set multiple tags to 1.

An External Debug reset clears the CLAIM tag bits to 0.

## Accessing the CTICLAIMSET

CTICLAIMSET can be accessed through the external debug interface:

Component	Offset
-----------	--------



CTI	0xFA0
-----	-------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTICONTROL, CTI Control register

The CTICONTROL characteristics are:

## Purpose

Controls whether the CTI is enabled.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RW

## Configuration

CTICONTROL is in the Debug power domain. Some or all RW fields of this register have defined reset values. These apply only on an External debug reset. The register is not affected by a Warm reset and is not affected by a Cold reset.

## Attributes

CTICONTROL is a 32-bit register.

## Field descriptions

The CTICONTROL 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	GLBEN

### Bits [31:1]

Reserved, RES0.

### GLBEN, bit [0]

Enables or disables the CTI mapping functions. Possible values of this field are:

GLBEN	Meaning
0	CTI mapping functions disabled.
1	CTI mapping functions enabled.

When the mapping functions are disabled, no new events are signaled on either output triggers or output channels. If a previously asserted output trigger has not been acknowledged, it remains asserted after the mapping functions are disabled. All output triggers are disabled by CTI reset.

If the ECT supports multicycle channel events any existing output channel events will be terminated.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the CTICONTROL

CTICONTROL can be accessed through the external debug interface:

Component	Offset
CTI	0x000

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIDEVAFF0, CTI Device Affinity register 0

The CTIDEVAFF0 characteristics are:

## Purpose

Copy of the low half of the PE [MPIDR\\_EL1](#) register that allows a debugger to determine which PE in a multiprocessor system the CTI component relates to.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIDEVAFF0 is in the Debug power domain.

Implementation of this register is OPTIONAL.

## Attributes

CTIDEVAFF0 is a 32-bit register.

## Field descriptions

The CTIDEVAFF0 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
																<a href="#">MPIDR_EL1 low half</a>															

### Bits [31:0]

[MPIDR\\_EL1](#) low half. Read-only copy of the low half of [MPIDR\\_EL1](#), as seen from the highest implemented Exception level.

## Accessing the CTIDEVAFF0

CTIDEVAFF0 can be accessed through the external debug interface:

Component	Offset
CTI	0xFA8

# CTIDEVAFF1, CTI Device Affinity register 1

The CTIDEVAFF1 characteristics are:

## Purpose

Copy of the high half of the PE [MPIDR\\_EL1](#) register that allows a debugger to determine which PE in a multiprocessor system the CTI component relates to.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIDEVAFF1 is in the Debug power domain.

Implementation of this register is OPTIONAL.

## Attributes

CTIDEVAFF1 is a 32-bit register.

## Field descriptions

The CTIDEVAFF1 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
																<a href="#">MPIDR_EL1 high half</a>															

### Bits [31:0]

[MPIDR\\_EL1](#) high half. Read-only copy of the high half of [MPIDR\\_EL1](#), as seen from the highest implemented Exception level.

## Accessing the CTIDEVAFF1

CTIDEVAFF1 can be accessed through the external debug interface:

Component	Offset
CTI	0xFAC

# CTIDEVARCH, CTI Device Architecture register

The CTIDEVARCH characteristics are:

## Purpose

Identifies the programmers' model architecture of the CTI component.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIDEVARCH is in the Debug power domain.

Implementation of this register is OPTIONAL.

## Attributes

CTIDEVARCH is a 32-bit register.

## Field descriptions

The CTIDEVARCH 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
ARCHITECT											PRESENT	REVISION				ARCHID															

### ARCHITECT, bits [31:21]

Defines the architecture of the component. For CTI, this is ARM Limited.

Bits [31:28] are the JEP106 continuation code, 0×4.

Bits [27:21] are the JEP106 ID code, 0×3B.

### PRESENT, bit [20]

When set to 1, indicates that the DEVARCH is present.

This field is 1 in ARMv8.

### REVISION, bits [19:16]

Defines the architecture revision. For architectures defined by ARM this is the minor revision.

For CTI, the revision defined by ARMv8 is 0×0.

All other values are reserved.

**ARCHID, bits [15:0]**

Defines this part to be an ARMv8 debug component. For architectures defined by ARM this is further subdivided.

For CTI:

- Bits [15:12] are the architecture version, 0x1.
- Bits [11:0] are the architecture part number, 0xA14.

This corresponds to CTI architecture version CTIv2.

## Accessing the CTIDEVARCH

CTIDEVARCH can be accessed through the external debug interface:

Component	Offset
CTI	0xFBC

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIDEVID, CTI Device ID register 0

The CTIDEVID characteristics are:

## Purpose

Describes the CTI component to the debugger.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIDEVID is in the Debug power domain.

## Attributes

CTIDEVID is a 32-bit register.

## Field descriptions

The CTIDEVID 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	INOUT	0	0			NUMCHAN					0	0			NUMTRIG				0	0	0		EXTMUXNUM			

### Bits [31:26]

Reserved, RES0.

### INOUT, bits [25:24]

Input/output options. Indicates presence of the input gate. If the CTM is not implemented, this field is RAZ.

INOUT	Meaning
00	<a href="#">CTIGATE</a> does not mask propagation of input events from external channels.
01	<a href="#">CTIGATE</a> masks propagation of input events from external channels.

All other values are reserved.

### Bits [23:22]

Reserved, RES0.

### NUMCHAN, bits [21:16]

Number of ECT channels implemented. IMPLEMENTATION DEFINED. For ARMv8, valid values are:



NUMCHAN	Meaning
000011	3 channels (0..2) implemented.
000100	4 channels (0..3) implemented.
000101	5 channels (0..4) implemented.
000110	6 channels (0..5) implemented.

and so on up to 100000, 32 channels (0..31) implemented.

All other values are reserved.

#### Bits [15:14]

Reserved, RES0.

#### NUMTRIG, bits [13:8]

Number of triggers implemented. IMPLEMENTATION DEFINED. This is one more than the index of the largest trigger, rather than the actual number of triggers.

For ARMv8, valid values are:

NUMTRIG	Meaning
000011	Up to 3 triggers (0..2) implemented.
001000	Up to 8 triggers (0..7) implemented.
001001	Up to 9 triggers (0..8) implemented.
001010	Up to 10 triggers (0..9) implemented.

and so on up to 100000, 32 triggers (0..31) implemented.

All other values are reserved. If the contains a Trace extension, this field must be at least 0b001000. There is no guarantee that any of the implemented triggers, including the highest numbered, are connected to any components.

#### Bits [7:5]

Reserved, RES0.

#### EXTMUXNUM, bits [4:0]

Number of multiplexors available on triggers. This value is used in conjunction with External Control register, [ASICCTL](#).

## Accessing the CTIDEVID

CTIDEVID can be accessed through the external debug interface:

Component	Offset
CTI	0xFC8

# CTIDEVID1, CTI Device ID register 1

The CTIDEVID1 characteristics are:

## Purpose

Reserved for future information about the CTI component to the debugger.  
This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIDEVID1 is in the Debug power domain.

## Attributes

CTIDEVID1 is a 32-bit register.

## Field descriptions

The CTIDEVID1 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	0

### Bits [31:0]

Reserved, RES0.

## Accessing the CTIDEVID1

CTIDEVID1 can be accessed through the external debug interface:

Component	Offset
CTI	0xFC4

# CTIDEVID2, CTI Device ID register 2

The CTIDEVID2 characteristics are:

## Purpose

Reserved for future information about the CTI component to the debugger.  
This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIDEVID2 is in the Debug power domain.

## Attributes

CTIDEVID2 is a 32-bit register.

## Field descriptions

The CTIDEVID2 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	0

### Bits [31:0]

Reserved, RES0.

## Accessing the CTIDEVID2

CTIDEVID2 can be accessed through the external debug interface:

Component	Offset
CTI	0xFC0

# CTIDEVTYPE, CTI Device Type register

The CTIDEVTYPE characteristics are:

## Purpose

Indicates to a debugger that this component is part of a PEs cross-trigger interface.  
This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIDEVTYPE is in the Debug power domain.  
Implementation of this register is OPTIONAL.

## Attributes

CTIDEVTYPE is a 32-bit register.

## Field descriptions

The CTIDEVTYPE 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	SUB				MAJOR			

### Bits [31:8]

Reserved, RES0.

### SUB, bits [7:4]

Subtype. Must read as 0x1 to indicate this is a component within a PE.

### MAJOR, bits [3:0]

Major type. Must read as 0x4 to indicate this is a cross-trigger component.

## Accessing the CTIDEVTYPE

CTIDEVTYPE can be accessed through the external debug interface:

Component	Offset
CTI	0xFCC



# CTIGATE, CTI Channel Gate Enable register

The CTIGATE characteristics are:

## Purpose

Determines whether events on channels propagate through the CTM to other ECT components, or from the CTM into the CTI.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RW

## Configuration

CTIGATE is in the Debug power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply only on an External debug reset. The register is not affected by a Warm reset and is not affected by a Cold reset.

## Attributes

CTIGATE is a 32-bit register.

## Field descriptions

The CTIGATE 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
GATE<x>, bit [x]																															

### GATE<x>, bit [x], for x = 0 to 31

Channel <x> gate enable.

Bits [31:N] are RAZ/WI. N is the number of ECT channels implemented as defined by the CTIDEVID.NUMCHAN field.

Possible values of this bit are:

GATE<x>	Meaning
0	Disable output and, if CTIDEVID.INOUT == 0b01, input channel <x> propagation.
1	Enable output and, if CTIDEVID.INOUT == 0b01, input channel <x> propagation.

If GATE[x] is set to 0, no new events will be propagated to the ECT, and if the ECT supports multicycle channel events any existing output channel events will be terminated.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the CTIGATE

CTIGATE can be accessed through the external debug interface:

Component	Offset
CTI	0x140

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIINEN<n>, CTI Input Trigger to Output Channel Enable registers, n = 0 - 31

The CTIINEN<n> characteristics are:

## Purpose

Enables the signaling of an event on output channels when input trigger event n is received by the CTI.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RW

## Configuration

CTIINEN<n> is in the Debug power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply only on an External debug reset. The register is not affected by a Warm reset and is not affected by a Cold reset.

If input trigger n is not connected, the behavior of CTIINEN<n> is IMPLEMENTATION DEFINED.

## Attributes

CTIINEN<n> is a 32-bit register.

## Field descriptions

The CTIINEN<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
																INEN<x>, bit [x]															

### INEN<x>, bit [x], for x = 0 to 31

Input trigger <n> to output channel <x> enable.

Bits [31:N] are RAZ/WI. N is the number of ECT channels implemented as defined by the [CTIDEVID.NUMCHAN](#) field.

Possible values of this bit are:

INEN<x>	Meaning
0	Input trigger <n> will not generate an event on output channel <x>.
1	Input trigger <n> will generate an event on output channel <x>.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the CTIINEN<n>

CTIINEN<n> can be accessed through the external debug interface:

Component	Offset
-----------	--------



CTI	$0 \times 020 + 4n$
-----	---------------------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIINTACK, CTI Output Trigger Acknowledge register

The CTIINTACK characteristics are:

## Purpose

Can be used to deactivate the output triggers.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	WO

A debugger must read [CTITRIGOUTSTATUS](#) to confirm that the output trigger has been acknowledged before generating any event that must be ordered after the write to CTIINTACK, such as a write to CTIAPPULSE to activate another trigger.

## Configuration

CTIINTACK is in the Debug power domain.

## Attributes

CTIINTACK is a 32-bit register.

## Field descriptions

The CTIINTACK 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
<a href="#">ACK&lt;n&gt;, bit [n]</a>																															

### ACK<n>, bit [n], for n = 0 to 31

Acknowledge for output trigger <n>.

Bits [31:N] are RAZ/WI. N is the number of CTI triggers implemented as defined by the [CTIDEVID](#).NUMTRIG field.

If any of the following is true, writes to ACK<n> are ignored:

- $n \geq$  [CTIDEVID](#).NUMTRIG, the number of implemented triggers.
- Output trigger n is not active.
- The channel mapping function output, as controlled by [CTIOUTEN<n>](#), is still active.

Otherwise, if any of the following are true, it is IMPLEMENTATION DEFINED whether writes to ACK<n> are ignored:

- Output trigger n is not implemented.
- Output trigger n is not connected.
- Output trigger n is self-acknowledging and does not require software acknowledge.

Otherwise, the behavior on writes to ACK<n> is as follows:

ACK<n>	Meaning
0	No effect
1	Deactivate the trigger.

## Accessing the CTIINTACK

CTIINTACK can be accessed through the external debug interface:

Component	Offset
CTI	0x010

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIITCTRL, CTI Integration mode Control register

The CTIITCTRL characteristics are:

## Purpose

Enables the CTI to switch from its default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RW

## Configuration

It is IMPLEMENTATION DEFINED whether CTIITCTRL is implemented in the Core power domain or in the Debug power domain. Some or all RW fields of this register have defined reset values, and:

- The register is not affected by a Warm reset.
- If the register is implemented in the Core power domain the reset values apply on a Cold reset, and the register is not affected by an External debug reset.
- If the register is implemented in the Debug power domain the reset values apply on an External debug reset, and the register is not affected by a Cold reset.

Implementation of this register is OPTIONAL.

## Attributes

CTIITCTRL is a 32-bit register.

## Field descriptions

The CTIITCTRL 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	IME

### Bits [31:1]

Reserved, RES0.

### IME, bit [0]

Integration mode enable. When IME == 1, the device reverts to an integration mode to enable integration testing or topology detection. The integration mode behavior is IMPLEMENTATION DEFINED.

IME	Meaning
0	Normal operation.
1	Integration mode enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the CTIITCTRL

CTIITCTRL can be accessed through the external debug interface:

Component	Offset
CTI	0xF00

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTILAR, CTI Lock Access Register

The CTILAR characteristics are:

## Purpose

Allows or disallows access to the CTI registers through a memory-mapped interface.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

Default
WO

## Configuration

CTILAR is in the Debug power domain.

If OPTIONAL memory-mapped access to the external debug interface is supported then an OPTIONAL Software Lock can be implemented as part of CoreSight compliance.

CTILAR ignores writes if the Software lock is not implemented and ignores writes for other accesses to the external debug interface.

The Software Lock provides a lock to prevent memory-mapped writes to the Cross-Trigger Interface registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Cross-Trigger Interface registers. It does not, and cannot, prevent all accidental or malicious damage.

Software uses CTILAR to set or clear the lock, and [CTILSR](#) to check the current status of the lock.

## Attributes

CTILAR is a 32-bit register.

## Field descriptions

The CTILAR 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
																KEY															

### KEY, bits [31:0]

Lock Access control. Writing the key value 0xC5ACCE55 to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface.

Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.

## Accessing the CTILAR

CTILAR can be accessed through a memory-mapped access to the external debug interface:

Component	Offset
CTI	0xFB0



# CTILSR, CTI Lock Status Register

The CTILSR characteristics are:

## Purpose

Indicates the current status of the Software Lock for CTI registers.  
This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

CTILSR is in the Debug power domain. Some or all RW fields of this register have defined reset values. These apply only on an External debug reset. The register is not affected by a Warm reset and is not affected by a Cold reset.

If OPTIONAL memory-mapped access to the external debug interface is supported then an OPTIONAL Software Lock can be implemented as part of CoreSight compliance.

CTILSR is RAZ if the Software Lock is not implemented and is RAZ for other accesses to the external debug interface.

The Software Lock provides a lock to prevent memory-mapped writes to the Cross-Trigger Interface registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Cross-Trigger Interface registers. It does not, and cannot, prevent all accidental or malicious damage.

Software uses [CTILAR](#) to set or clear the lock, and CTILSR to check the current status of the lock.

## Attributes

CTILSR is a 32-bit register.

## Field descriptions

The CTILSR 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	nTT	SLK	SLI

### Bits [31:3]

Reserved, RES0.

### nTT, bit [2]

Not thirty-two bit access required. RAZ.

### SLK, bit [1]

Software Lock status for this component. For an access to LSR that is not a memory-mapped access, or when the Software Lock is not implemented, this field is RES0.



For memory-mapped accesses when the Software Lock is implemented, possible values of this field are:

SLK	Meaning
0	Lock clear. Writes are permitted to this component's registers.
1	Lock set. Writes to this component's registers are ignored, and reads have no side effects.

When this register has an architecturally-defined reset value, this field resets to 1.

### SLI, bit [0]

Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is IMPLEMENTATION DEFINED. Permitted values are:

SLI	Meaning
0	Software Lock not implemented or not memory-mapped access.
1	Software Lock implemented and memory-mapped access.

## Accessing the CTILSR

CTILSR can be accessed through a memory-mapped access to the external debug interface:

Component	Offset
CTI	0xFB4

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIOUTEN<n>, CTI Input Channel to Output Trigger Enable registers, n = 0 - 31

The CTIOUTEN<n> characteristics are:

## Purpose

Defines which input channels generate output trigger n.  
This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RW

## Configuration

CTIOUTEN<n> is in the Debug power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply only on an External debug reset. The register is not affected by a Warm reset and is not affected by a Cold reset.  
If output trigger n is not connected, the behavior of CTIOUTEN<n> is IMPLEMENTATION DEFINED.

## Attributes

CTIOUTEN<n> is a 32-bit register.

## Field descriptions

The CTIOUTEN<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
OUTEN<x>, bit [x]																															

### OUTEN<x>, bit [x], for x = 0 to 31

Input channel <x> to output trigger <n> enable.  
Bits [31:N] are RAZ/WI. N is the number of ECT channels implemented as defined by the [CTIDEVID.NUMCHAN](#) field.  
Possible values of this bit are:

OUTEN<x>	Meaning
0	An event on input channel <x> will not cause output trigger <n> to be asserted.
1	An event on input channel <x> will cause output trigger <n> to be asserted.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the CTIOUTEN<n>

CTIOUTEN<n> can be accessed through the external debug interface:

Component	Offset
CTI	0x0A0 + 4n

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIPIDR0, CTI Peripheral Identification Register 0

The CTIPIDR0 characteristics are:

## Purpose

Provides information to identify a CTI component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIPIDR0 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

CTIPIDR0 is a 32-bit register.

## Field descriptions

The CTIPIDR0 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	PART_0									

### Bits [31:8]

Reserved, RES0.

### PART\_0, bits [7:0]

Part number, least significant byte.

## Accessing the CTIPIDR0

CTIPIDR0 can be accessed through the external debug interface:

Component	Offset
CTI	0xFE0



# CTIPIDR1, CTI Peripheral Identification Register 1

The CTIPIDR1 characteristics are:

## Purpose

Provides information to identify a CTI component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIPIDR1 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

CTIPIDR1 is a 32-bit register.

## Field descriptions

The CTIPIDR1 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	DES_0			PART_1				

### Bits [31:8]

Reserved, RES0.

### DES\_0, bits [7:4]

Designer, least significant nibble of JEP106 ID code. For ARM Limited, this field is 0b1011.

### PART\_1, bits [3:0]

Part number, most significant nibble.

## Accessing the CTIPIDR1

CTIPIDR1 can be accessed through the external debug interface:

Component	Offset
-----------	--------

CTI	0xFE4
-----	-------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIPIDR2, CTI Peripheral Identification Register 2

The CTIPIDR2 characteristics are:

## Purpose

Provides information to identify a CTI component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIPIDR2 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

CTIPIDR2 is a 32-bit register.

## Field descriptions

The CTIPIDR2 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	REVISION	JEDEC	DES	1				

### Bits [31:8]

Reserved, RES0.

### REVISION, bits [7:4]

Part major revision. Parts can also use this field to extend Part number to 16-bits.

### JEDEC, bit [3]

RAO. Indicates a JEP106 identity code is used.

### DES\_1, bits [2:0]

Designer, most significant bits of JEP106 ID code. For ARM Limited, this field is 0b011.



## Accessing the CTIPIDR2

CTIPIDR2 can be accessed through the external debug interface:

Component	Offset
CTI	0xFE8

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIPIDR3, CTI Peripheral Identification Register 3

The CTIPIDR3 characteristics are:

## Purpose

Provides information to identify a CTI component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIPIDR3 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

CTIPIDR3 is a 32-bit register.

## Field descriptions

The CTIPIDR3 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	7	6	5	4	3	2	1	0
																								REVAND				CMOD			

### Bits [31:8]

Reserved, RES0.

### REVAND, bits [7:4]

Part minor revision. Parts using [CTIPIDR2](#).REVISION as an extension to the Part number must use this field as a major revision number.

### CMOD, bits [3:0]

Customer modified. Indicates someone other than the Designer has modified the component.

## Accessing the CTIPIDR3

CTIPIDR3 can be accessed through the external debug interface:

Component	Offset
-----------	--------

CTI	0xFEC
-----	-------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTIPIDR4, CTI Peripheral Identification Register 4

The CTIPIDR4 characteristics are:

## Purpose

Provides information to identify a CTI component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTIPIDR4 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

CTIPIDR4 is a 32-bit register.

## Field descriptions

The CTIPIDR4 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	SIZE				DES_2			

### Bits [31:8]

Reserved, RES0.

### SIZE, bits [7:4]

Size of the component. RAZ. Log2 of the number of 4KB pages from the start of the component to the end of the component ID registers.

### DES\_2, bits [3:0]

Designer, JEP106 continuation code, least significant nibble. For ARM Limited, this field is 0b0100.

## Accessing the CTIPIDR4

CTIPIDR4 can be accessed through the external debug interface:

Component	Offset
-----------	--------

CTI	0xFD0
-----	-------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# CTITRIGINSTATUS, CTI Trigger In Status register

The CTITRIGINSTATUS characteristics are:

## Purpose

Provides the status of the trigger inputs.  
This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTITRIGINSTATUS is in the Debug power domain.

## Attributes

CTITRIGINSTATUS is a 32-bit register.

## Field descriptions

The CTITRIGINSTATUS 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
																TRIN<n>, bit [n]															

### TRIN<n>, bit [n], for n = 0 to 31

Trigger input <n> status.  
Bits [31:N] are RAZ. N is the number of CTI triggers implemented as defined by the [CTIDEVID](#).NUMTRIG field.  
Possible values of this bit are:

TRIN<n>	Meaning
0	Input trigger n is inactive.
1	Input trigger n is active.

Not implemented and not-connected input triggers are always inactive.  
It is IMPLEMENTATION DEFINED whether an input trigger that does not support multicyle events can be observed as active.

## Accessing the CTITRIGINSTATUS

CTITRIGINSTATUS can be accessed through the external debug interface:

Component	Offset
CTI	0x130



# CTITRIGOUTSTATUS, CTI Trigger Out Status register

The CTITRIGOUTSTATUS characteristics are:

## Purpose

Provides the raw status of the trigger outputs, after processing by any IMPLEMENTATION DEFINED trigger interface logic. For output triggers that are self-acknowledging, this is only meaningful if the CTI implements multicycle channel events.

This register is part of the Cross-Trigger Interface registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

CTITRIGOUTSTATUS is in the Debug power domain.

## Attributes

CTITRIGOUTSTATUS is a 32-bit register.

## Field descriptions

The CTITRIGOUTSTATUS 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
TROUT<n>, bit [n]																															

### TROUT<n>, bit [n], for n = 0 to 31

Trigger output <n> status.

Bits [31:N] are RAZ, where N is the value of the [CTIDEVID.NUMTRIG](#) field.

If  $n < N$ , and output trigger <n> is implemented and connected, and either the trigger is not self-acknowledging or the CTI implements multicycle channel events, then permitted values for TROUT<n> are:

TROUT<n>	Meaning
0	Output trigger n is inactive.
1	Output trigger n is active.

Otherwise when  $n < N$  it is IMPLEMENTATION DEFINED whether TROUT<n> behaves as described here or is RAZ.

## Accessing the CTITRIGOUTSTATUS

CTITRIGOUTSTATUS can be accessed through the external debug interface:

Component	Offset
CTI	0x134





# CounterID<n>, Counter ID registers, n = 0 - 11

The CounterID<n> characteristics are:

## Purpose

IMPLEMENTATION DEFINED identification registers 0 to 11 for the memory-mapped Generic Timer.

This register is part of the Generic Timer registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

These registers must be implemented, as RO registers, in every implemented Generic Timer memory-mapped frame.

For the CNTCTLBase frame, in a system that recognizes two Security states these registers are accessible by both Secure and Non-secure accesses.

For the CNTControlBase frame, in a system that supports Secure and Non-secure memory maps the frame is implemented only in the Secure memory map, meaning these registers are implemented only in the Secure memory map.

For the CNTBaseN frames, '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 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.

## Configuration

The power domain of CounterID<n> is IMPLEMENTATION DEFINED.

For more information see 'Power and reset domains for the system level implementation of the Generic Timer' in Chapter I1 of the ARMv8 ARM.

These registers are implemented independently in each of the implemented Generic Timer memory-mapped frames.

If the implementation of the Counter ID registers requires an architecture version, the value for this version of the ARM Generic Timer is version 0.

The Counter ID registers can be implemented as a set of CoreSight ID registers, comprising Peripheral ID Registers and Component ID Registers. An implementation of these registers for the Generic Timer must use a Component class value of 0xF.

## Attributes

CounterID<n> is a 32-bit register.

## Field descriptions

The CounterID<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
IMPLEMENTATION DEFINED																															

**IMPLEMENTATION DEFINED, bits [31:0]**

IMPLEMENTATION DEFINED.

**Accessing the CounterID<n>**

CounterID&lt;n&gt; can be accessed through its memory-mapped interface:

Component	Frame	Offset
Timer	CNTControlBase	$0 \times \text{FD0} + 4n$
Timer	CNTReadBase	$0 \times \text{FD0} + 4n$
Timer	CNTBaseN	$0 \times \text{FD0} + 4n$
Timer	CNTELOBaseN	$0 \times \text{FD0} + 4n$
Timer	CNTCTLBase	$0 \times \text{FD0} + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DBGAUTHSTATUS\_EL1, Debug Authentication Status register

The DBGAUTHSTATUS\_EL1 characteristics are:

## Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for debug.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

External register DBGAUTHSTATUS\_EL1 is architecturally mapped to AArch64 System register [DBGAUTHSTATUS\\_EL1](#).

External register DBGAUTHSTATUS\_EL1 is architecturally mapped to AArch32 System register [DBGAUTHSTATUS](#).

DBGAUTHSTATUS\_EL1 is in the Debug power domain.

## Attributes

DBGAUTHSTATUS\_EL1 is a 32-bit register.

## Field descriptions

The DBGAUTHSTATUS\_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	SNID	SID	NSNID	NSID				

### Bits [31:8]

Reserved, RES0.

### SNID, bits [7:6]

Secure non-invasive debug. Possible values of this field are:

SNID	Meaning
00	Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state.
10	Implemented and disabled. ExternalSecureNoninvasiveDebugEnabled() == FALSE.
11	Implemented and enabled. ExternalSecureNoninvasiveDebugEnabled() == TRUE.

Other values are reserved.

### SID, bits [5:4]

Secure invasive debug. Possible values of this field are:

SID	Meaning
00	Not implemented. EL3 is not implemented and the implemented Security state is Non-secure state.
10	Implemented and disabled. ExternalSecureInvasiveDebugEnabled() == FALSE.
11	Implemented and enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.

Other values are reserved.

### NSNID, bits [3:2]

Non-secure non-invasive debug. Possible values of this field are:

NSNID	Meaning
00	Not implemented. EL3 is not implemented and the implemented Security state is Secure state.
10	Implemented and disabled. ExternalNoninvasiveDebugEnabled() == FALSE.
11	Implemented and enabled. ExternalNoninvasiveDebugEnabled() == TRUE.

Other values are reserved.

### NSID, bits [1:0]

Non-secure invasive debug. Possible values of this field are:

NSID	Meaning
00	Not implemented. EL3 is not implemented and the implemented Security state is Secure state.
10	Implemented and disabled. ExternalInvasiveDebugEnabled() == FALSE.
11	Implemented and enabled. ExternalInvasiveDebugEnabled() == TRUE.

Other values are reserved.

## Accessing the DBGAUTHSTATUS\_EL1

DBGAUTHSTATUS\_EL1 can be accessed through the external debug interface:

Component	Offset
Debug	0×FB8

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
Error	Error	Error	Error	RO	RW

## Configuration

External register DBGBCR<n>\_EL1 is architecturally mapped to AArch64 System register [DBGBCR<n>\\_EL1](#).

External register DBGBCR<n>\_EL1 is architecturally mapped to AArch32 System register [DBGBCR<n>](#).

DBGBCR<n>\_EL1 is in the Core power domain. RW 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.

If breakpoint n is not implemented then this register is unallocated.

## 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		

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 <a href="#">CONTEXTIDR_EL1</a> match (introduced in ARMv8.1).
0111	Linked <a href="#">CONTEXTIDR_EL1</a> 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 <a href="#">CONTEXTIDR_EL2</a> match (introduced in ARMv8.1).
1101	Linked <a href="#">CONTEXTIDR_EL2</a> 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>\\_EL1](#) is the address of an instruction.

001

Match Context ID. [DBGBCR<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](#).

010

Mismatch address. [DBGBCR<n>\\_EL1](#) is the address of an instruction to be stepped.

011

Match [CONTEXTIDR\\_EL1](#). [DBGBCR<n>\\_EL1](#).ContextID is a Context ID compared against [CONTEXTIDR\\_EL1](#).

100

Match VMID. [DBGBCR<n>\\_EL1](#).VMID is a VMID compared against [VTTBR\\_EL2](#).VMID.

101

Match VMID and Context ID. [DBGBCR<n>\\_EL1](#).ContextID is a Context ID compared against [CONTEXTIDR\\_EL1](#), and [DBGBCR<n>\\_EL1](#).VMID is a VMID compared against [VTTBR\\_EL2](#).VMID.

110

Match [CONTEXTIDR\\_EL2](#). [DBGBCR<n>\\_EL1](#).ContextID2 is a Context ID compared against [CONTEXTIDR\\_EL2](#).

111

Match Full Context ID. [DBGBCR<n>\\_EL1](#).ContextID is compared against [CONTEXTIDR\\_EL1](#), and [DBGBCR<n>\\_EL1](#).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>\_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 [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.

## 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 in either AArch32 or AArch64 state, the permitted values are:

BAS	Match instruction at	Constraint for debuggers
0011	<a href="#">DBGBVR&lt;n&gt;_EL1</a>	Use for T32 instructions.
1100	<a href="#">DBGBVR&lt;n&gt;_EL1+2</a>	Use for T32 instructions.
1111	<a href="#">DBGBVR&lt;n&gt;_EL1</a>	Use for A64 and A32 instructions.

All other values are reserved.

For more information, 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	<a href="#">DBGBVR&lt;n&gt;_EL1</a>	Use for stepping T32 instructions.
1100	<a href="#">DBGBVR&lt;n&gt;_EL1+2</a>	Use for stepping T32 instructions.
1111	<a href="#">DBGBVR&lt;n&gt;_EL1</a>	Use for stepping A64 and A32 instructions.

For more information, 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

DBGBCR<n>\_EL1 can be accessed through the external debug interface:

Component	Offset
Debug	$0 \times 408 + 16n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
Error	Error	Error	Error	RO	RW

## Configuration

External register DBGBVR<n>\_EL1 is architecturally mapped to AArch64 System register [DBGBVR<n>\\_EL1](#).

External register DBGBVR<n>\_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGBVR<n>](#).

External register DBGBVR<n>\_EL1 bits [63:32] are architecturally mapped to AArch32 System register [DBGXVR<n>](#).

DBGBVR<n>\_EL1 is in the Core power domain. RW 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.

If breakpoint n is not implemented then this register is unallocated.

## Attributes

How this register is interpreted depends on the value of [DBGBCR<n>\\_EL1](#).BT.

- When [DBGBCR<n>\\_EL1](#).BT is 0b0x0x, 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==0b0x0x:

63	62	61	60	59	58	57	56	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 ARMv8.2:

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]

If the address is being matched in an AArch64 stage 1 translation regime:

- This field contains bits[48:2] of the address 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.

If the address is being matched in an AArch32 stage 1 translation regime, the first 20 bits of this field are RES0, and the rest of the field contains bits[31:2] of the address 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>\_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](#) and [CONTEXTIDR\\_EL1](#) 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, [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 ARMv8.2 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 ARMv8.2 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]

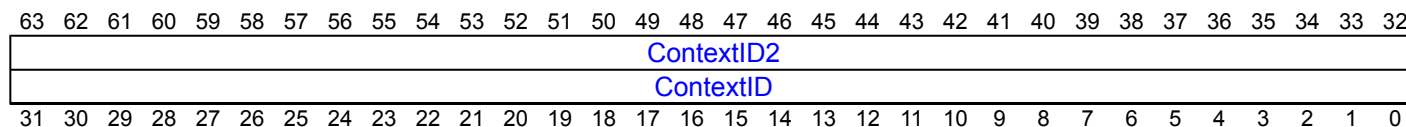
Context ID value for comparison against [CONTEXTIDR\\_EL2](#).

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:



## 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

DBGBVR<n>\_EL1[31:0] can be accessed through the external debug interface:

Component	Offset
Debug	0x400 + 16n

DBGBVR<n>\_EL1[63:32] can be accessed through the external debug interface:

Component	Offset
Debug	0x404 + 16n

# DBGCLAIMCLR\_EL1, Debug Claim Tag Clear register

The DBGCLAIMCLR\_EL1 characteristics are:

## Purpose

Used by software to read the values of the CLAIM tag bits, and to clear these bits to 0.

The architecture does not define any functionality for the CLAIM tag bits.

---

### Note

CLAIM tags are typically used for communication between the debugger and target software.

---

Used in conjunction with the [DBGCLAIMSET\\_EL1](#) register.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
Error	Error	Error	RO	RW

## Configuration

External register DBGCLAIMCLR\_EL1 is architecturally mapped to AArch64 System register [DBGCLAIMCLR\\_EL1](#).

External register DBGCLAIMCLR\_EL1 is architecturally mapped to AArch32 System register [DBGCLAIMCLR](#).

DBGCLAIMCLR\_EL1 is in the Core power domain.

See the CLAIM field description for the effect of a Cold reset on the value returned by this register. This register is not affected by a Warm reset, and is not affected by an External debug reset.

An implementation must include 8 CLAIM tag bits.

## Attributes

DBGCLAIMCLR\_EL1 is a 32-bit register.

## Field descriptions

The DBGCLAIMCLR\_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								

[CLAIM](#)

### Bits [31:8]

Reserved, RAZ/SBZ. Software can rely on these bits reading as zero, and must use a should-be-zero policy on writes. Implementations must ignore writes.

**CLAIM, bits [7:0]**

Read or clear CLAIM tag bits. Reading this field returns the current value of the CLAIM tag bits.

Writing a 1 to one of these bits clears the corresponding CLAIM tag bit to 0. This is an indirect write to the CLAIM tag bits. A single write operation can clear multiple CLAIM tag bits to 0.

Writing 0 to one of these bits has no effect.

A cold reset clears the CLAIM tag bits to 0.

**Accessing the DBGCLAIMCLR\_EL1**

DBGCLAIMCLR\_EL1 can be accessed through the external debug interface:

Component	Offset
Debug	0xFA4

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# DBGCLAIMSET\_EL1, Debug Claim Tag Set register

The DBGCLAIMSET\_EL1 characteristics are:

## Purpose

Used by software to set the CLAIM tag bits to 1.

The architecture does not define any functionality for the CLAIM tag bits.

---

### Note

CLAIM tags are typically used for communication between the debugger and target software.

---

Used in conjunction with the [DBGCLAIMCLR\\_EL1](#) register.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
Error	Error	Error	RO	RW

## Configuration

External register DBGCLAIMSET\_EL1 is architecturally mapped to AArch64 System register [DBGCLAIMSET\\_EL1](#).

External register DBGCLAIMSET\_EL1 is architecturally mapped to AArch32 System register [DBGCLAIMSET](#).

DBGCLAIMSET\_EL1 is in the Core power domain.

An implementation must include 8 CLAIM tag bits.

## Attributes

DBGCLAIMSET\_EL1 is a 32-bit register.

## Field descriptions

The DBGCLAIMSET\_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								

[CLAIM](#)

### Bits [31:8]

Reserved, RAZ/SBZ. Software can rely on these bits reading as zero, and must use a should-be-zero policy on writes. Implementations must ignore writes.

### CLAIM, bits [7:0]

Set CLAIM tag bits. RAO.

Writing a 1 to one of these bits sets the corresponding CLAIM tag bit to 1. This is an indirect write to the CLAIM tag bits. A single write operation can set multiple CLAIM tag bits to 1.

Writing 0 to one of these bits has no effect.

A cold reset clears the CLAIM tag bits to 0.

## Accessing the DBGCLAIMSET\_EL1

DBGCLAIMSET\_EL1 can be accessed through the external debug interface:

Component	Offset
Debug	0xFA0

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DBGDTRRX\_EL0, Debug Data Transfer Register, Receive

The DBGDTRRX\_EL0 characteristics are:

## Purpose

Transfers data from an external debugger to the PE. For example, it is used by a debugger transferring commands and data to a debug target. It is a component of the Debug Communications Channel.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
Error	Error	Error	RO	RW

If [EDSCR.ITE](#) == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any operation issued by a DTR access in memory access mode that has not completed execution is CONSTRAINED UNPREDICTABLE, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an UNKNOWN state.

## Configuration

External register DBGDTRRX\_EL0 is architecturally mapped to AArch64 System register [DBGDTRRX\\_EL0](#).

External register DBGDTRRX\_EL0 is architecturally mapped to AArch32 System register [DBGDTRRXint](#).

DBGDTRRX\_EL0 is in the Core power domain. RW 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.

## Attributes

DBGDTRRX\_EL0 is a 32-bit register.

## Field descriptions

The DBGDTRRX\_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
Update DTRRX																															

### Bits [31:0]

Update DTRRX.

If RXfull is set to 0, then writes to this register update the value in DTRRX and set RXfull to 1.

Reads of this register return the last value written to DTRRX and do not change RXfull.

For the full behavior of the Debug Communications Channel, see 'The Debug Communication Channel and Instruction Transfer Register' in the ARM ARM, chapter H4.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the DBGDTRRX\_EL0

DBGDTRRX\_EL0 can be accessed through the external debug interface:

Component	Offset
Debug	0x080

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# DBGDTRTX\_EL0, Debug Data Transfer Register, Transmit

The DBGDTRTX\_EL0 characteristics are:

## Purpose

Transfers data from the PE to an external debugger. For example, it is used by a debug target to transfer data to the debugger. It is a component of the Debug Communication Channel.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
Error	Error	Error	RO	RW

If [EDSCR.ITE](#) == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any operation issued by a DTR access in memory access mode that has not completed execution is CONSTRAINED UNPREDICTABLE, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an UNKNOWN state.

## Configuration

External register DBGDTRTX\_EL0 is architecturally mapped to AArch64 System register [DBGDTRTX\\_EL0](#).

External register DBGDTRTX\_EL0 is architecturally mapped to AArch32 System register [DBGDTRTXint](#).

DBGDTRTX\_EL0 is in the Core power domain. RW 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.

## Attributes

DBGDTRTX\_EL0 is a 32-bit register.

## Field descriptions

The DBGDTRTX\_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
Return DTRTX																															

### Bits [31:0]

Return DTRTX.

If TXfull is set to 1, then reads of this register return the value in DTRTX and clear TXfull to 0.

Writes of this register update the value in DTRTX and do not change TXfull.

For the full behavior of the Debug Communications Channel, see 'The Debug Communication Channel and Instruction Transfer Register' in the ARM ARM, chapter H4.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the DBGDTRTX\_EL0

DBGDTRTX\_EL0 can be accessed through the external debug interface:

Component	Offset
Debug	0x08C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
Error	Error	Error	Error	RO	RW

## Configuration

External register DBGWCR<n>\_EL1 is architecturally mapped to AArch64 System register [DBGWCR<n>\\_EL1](#).

External register DBGWCR<n>\_EL1 is architecturally mapped to AArch32 System register [DBGWCR<n>](#).

DBGWCR<n>\_EL1 is in the Core power domain. RW 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.

If breakpoint n is not implemented then this register is unallocated.

## 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					<b>MASK</b>		0	0	0	<b>WT</b>		<b>LBN</b>		<b>SSC</b>	<b>HMC</b>					<b>BAS</b>					<b>LSC</b>	<b>PAC</b>	<b>E</b>		

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 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
xxxxxx1	Match byte at <a href="#">DBGWVR&lt;n&gt;_EL1</a>
xxxxxx1x	Match byte at <a href="#">DBGWVR&lt;n&gt;_EL1</a> +1
xxxxx1xx	Match byte at <a href="#">DBGWVR&lt;n&gt;_EL1</a> +2
xxxx1xxx	Match byte at <a href="#">DBGWVR&lt;n&gt;_EL1</a> +3

In cases where [DBGWVR<n>\\_EL1](#) addresses a double-word:



BAS	Description, if DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at DBGWVR<n>_EL1+4
xx1xxxxx	Match byte at DBGWVR<n>_EL1+5
x1xxxxxx	Match byte at DBGWVR<n>_EL1+6
1xxxxxxx	Match byte at DBGWVR<n>_EL1+7

If DBGWVR<n>\_EL1[2] == 1, only BAS[3:0] is used. ARM deprecates setting DBGWVR<n>\_EL1[2] == 1.

The valid values for BAS are non-zero binary number all of whose set bits are contiguous. All other values are reserved and must not be used by software. See 'Reserved DBGWCR<n>.BAS values' 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 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

DBGWCR<n>\_EL1 can be accessed through the external debug interface:

Component	Offset
Debug	0x808 + 16n

# 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.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EDAD	SLK	Default
Error	Error	Error	Error	RO	RW

## Configuration

External register DBGWVR<n>\_EL1 is architecturally mapped to AArch64 System register [DBGWVR<n>\\_EL1](#).

External register DBGWVR<n>\_EL1 bits [31:0] are architecturally mapped to AArch32 System register [DBGWVR<n>](#).

DBGWVR<n>\_EL1 is in the Core power domain. RW 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.

If breakpoint n is not implemented then this register is unallocated.

## 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 ARMv8.2:

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

DBGWVR<n>\_EL1[31:0] can be accessed through the external debug interface:

Component	Offset
Debug	$0 \times 800 + 16n$

DBGWVR<n>\_EL1[63:32] can be accessed through the external debug interface:

Component	Offset
Debug	$0 \times 804 + 16n$

# EDAA32PFR, External Debug AArch32 Processor Feature Register

The EDAA32PFR characteristics are:

## Purpose

Provides information about implemented PE features.

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.

## Usage constraints

This register is accessible as follows:

Off	DLK	Default
IMP DEF	IMP DEF	RO

## Configuration

It is IMPLEMENTATION DEFINED whether EDAA32PFR is implemented in the Core power domain or in the Debug power domain.

EDAA32PFR is only accessible in an implementation that only supports execution in AA32 state. If AArch64 state is supported at any Exception level, EDAA32PFR is RES0.

## Attributes

EDAA32PFR is a 64-bit register.

## Field descriptions

The EDAA32PFR 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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EL3				EL2				PMSA				VMSA			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:16]

Reserved, RES0.

### EL3, bits [15:12]

AArch32 EL3 Exception level handling. Defined values are:

EL3	Meaning
0000	EL3 is not implemented.
0001	EL3 can be executed in AArch32 state only.

When the value of [EDPFR.EL3](#) is non-zero, this field must be 0000.

All other values are reserved.

**Note**

[EDPFR](#).{EL1, EL0} indicate whether EL1 and EL0 can only be executed in AArch32 state.

**EL2, bits [11:8]**

AArch32 EL2 Exception level handling. Defined values are:

EL2	Meaning
0000	EL2 is not implemented.
0001	EL2 can be executed in AArch32 state only.

When the value of [EDPFR](#).EL2 is non-zero, this field must be 0000.

All other values are reserved.

**Note**

[EDPFR](#).{EL1, EL0} indicate whether EL1 and EL0 can only be executed in AArch32 state.

**PMSA, bits [7:4]**

Indicates support for a PMSA. Defined values are:

PMSA	Meaning
0000	PMSA not supported.
0100	Support for an ARMv8-R PMSAv8-32.

All other values are reserved. In ARMv8-A, the only permitted value is 0000.

**VMSA, bits [3:0]**

Indicates support for a VMSA. When the PMSA field is nonzero, determines support for a VMSA. When the PMSA field is 0000, VMSA is supported. Defined values are:

VMSA	Meaning
0000	VMSA not supported.

All other values are reserved. In ARMv8-A, the only permitted value is 0000.

**Accessing the EDAA32PFR**

EDAA32PFR can be accessed through the external debug interface:

Component	Offset
Debug	0xD60

# EDACR, External Debug Auxiliary Control Register

The EDACR characteristics are:

## Purpose

Allows implementations to support IMPLEMENTATION DEFINED controls.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
IMP DEF	IMP DEF	IMP DEF	RO	RW

## Configuration

It is IMPLEMENTATION DEFINED whether EDACR is implemented in the Core power domain or in the Debug power domain. RW fields in this register reset to architecturally UNKNOWN values, and:

- The register is not affected by a Warm reset.
- If the register is implemented in the Core power domain the reset values apply on a Cold reset, and the register is not affected by an External debug reset.
- If the register is implemented in the Debug power domain the reset values apply on an External debug reset, and the register is not affected by a Cold reset.

Changing this register from its reset value causes IMPLEMENTATION DEFINED behavior, including possible deviation from the architecturally-defined behavior.

If the EDACR contains any control bits that must be preserved over power down, then these bits must be accessible by the external debug interface when [OSLSR\\_ELI](#).OSLK == 1 (OS lock is locked) and when the Core is powered off.

## Attributes

EDACR is a 32-bit register.

## Field descriptions

The EDACR 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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the EDACR

EDACR can be accessed through the external debug interface:

Component	Offset
-----------	--------

Debug	0x094
-------	-------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

## EDCIDR0, External Debug Component Identification Register 0

The EDCIDR0 characteristics are:

## Purpose

Provides information to identify an external debug component.

For more information see 'About the Component identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

<b>Default</b>
RO

## Configuration

EDCIDR0 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

EDCIDR0 is a 32-bit register.

## Field descriptions

The EDCIDR0 bit assignments are:

[illegible]**Bits [31:8]**

Reserved, RES0.

**PRMBL\_0, bits [7:0]**

Preamble. Must read as 0x0D.

## Accessing the EDCIDR0

EDCIDR0 can be accessed through the external debug interface:

Component	Offset
Debug	0xFF0





# EDCIDR1, External Debug Component Identification Register 1

The EDCIDR1 characteristics are:

## Purpose

Provides information to identify an external debug component.

For more information see 'About the Component identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDCIDR1 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

EDCIDR1 is a 32-bit register.

## Field descriptions

The EDCIDR1 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	CLASS					PRMBL_1		

### Bits [31:8]

Reserved, RES0.

### CLASS, bits [7:4]

Component class. Reads as 0x9, debug component.

### PRMBL\_1, bits [3:0]

Preamble. RAZ.

## Accessing the EDCIDR1

EDCIDR1 can be accessed through the external debug interface:

Component	Offset
-----------	--------

Debug	0xFF4
-------	-------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

## EDCIDR2, External Debug Component Identification Register 2

The EDCIDR2 characteristics are:

## Purpose

Provides information to identify an external debug component.

For more information see 'About the Component identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

<b>Default</b>
RO

## Configuration

EDCIDR2 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

EDCIDR2 is a 32-bit register.

## Field descriptions

The EDCIDR2 bit assignments are:

[illegible]**Bits [31:8]**

Reserved, RES0.

**PRMBL\_2, bits [7:0]**

Preamble. Must read as 0x05.

## Accessing the EDCIDR2

EDCIDR2 can be accessed through the external debug interface:

Component	Offset
Debug	0xFF8



# EDCIDR3, External Debug Component Identification Register 3

The EDCIDR3 characteristics are:

## Purpose

Provides information to identify an external debug component.

For more information see 'About the Component identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDCIDR3 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

EDCIDR3 is a 32-bit register.

## Field descriptions

The EDCIDR3 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	PRMBL_3									

### Bits [31:8]

Reserved, RES0.

### PRMBL\_3, bits [7:0]

Preamble. Must read as 0xB1.

## Accessing the EDCIDR3

EDCIDR3 can be accessed through the external debug interface:

Component	Offset
Debug	0xFFC



# EDCIDSR, External Debug Context ID Sample Register

The EDCIDSR characteristics are:

## Purpose

Contains the sampled value of the Context ID, captured on reading [EDPCSR](#)[31:0].

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	Default
Error	Error	Error	RO

## Configuration

EDCIDSR 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 if the OPTIONAL PC Sample-based Profiling Extension is implemented but not with ARMv8.2-PCSample. If ARMv8.2-PCSample is implemented, this register is RES0 and the architecture defines the functionality in [PMCID1SR](#) and [PMCID2SR](#).

## Attributes

EDCIDSR is a 32-bit register.

## Field descriptions

The EDCIDSR bit assignments are:

### When ARMv8.2-PCSample is not 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
CONTEXTIDR																															

### CONTEXTIDR, bits [31:0]

Context ID. The value of CONTEXTIDR that is associated with the most recent [EDPCSR](#) sample.

- If EL1 is using AArch64, then the Context ID is held in [CONTEXTIDR\\_EL1](#).
- If EL1 is using AArch32, then the Context ID is held in [CONTEXTIDR](#).
- If EL3 is implemented and is using AArch32, then [CONTEXTIDR](#) is a banked register, and EDCIDSR samples the current banked copy of [CONTEXTIDR](#) for the Security state that is associated with the most recent EDPCSR sample.
- If [EDPCSR.NS](#)==0, this field is set to an UNKNOWN value.

Because the value written to EDCIDSR is an indirect read of CONTEXTIDR, therefore it is CONSTRAINED UNPREDICTABLE whether EDCIDSR is set to the original or new value if a read of [EDPCSR](#) samples:

- An instruction that writes to [CONTEXTIDR\\_EL1](#).
- The next Context synchronization event.
- Any instruction executed between these two instructions.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.



When ARMv8.2-PCSample is implemented:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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

Bits [31:0]

Reserved, RES0.

Accessing the EDCIDSR

EDCIDSR can be accessed through the external debug interface:

Component	Offset
Debug	0x0A4

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDDEVAFF0, External Debug Device Affinity register 0

The EDDEVAFF0 characteristics are:

## Purpose

Copy of the low half of the PE [MPIDR\\_EL1](#) register that allows a debugger to determine which PE in a multiprocessor system the external debug component relates to.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDDEVAFF0 is in the Debug power domain.

Implementation of this register is OPTIONAL.

## Attributes

EDDEVAFF0 is a 32-bit register.

## Field descriptions

The EDDEVAFF0 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
																<a href="#">MPIDR_EL1 low half</a>															

### Bits [31:0]

[MPIDR\\_EL1](#) low half. Read-only copy of the low half of [MPIDR\\_EL1](#), as seen from the highest implemented Exception level.

## Accessing the EDDEVAFF0

EDDEVAFF0 can be accessed through the external debug interface:

Component	Offset
Debug	0xFA8

# EDDEVAFF1, External Debug Device Affinity register 1

The EDDEVAFF1 characteristics are:

## Purpose

Copy of the high half of the PE [MPIDR\\_EL1](#) register that allows a debugger to determine which PE in a multiprocessor system the external debug component relates to.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDDEVAFF1 is in the Debug power domain.

Implementation of this register is OPTIONAL.

## Attributes

EDDEVAFF1 is a 32-bit register.

## Field descriptions

The EDDEVAFF1 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
																<a href="#">MPIDR_EL1 high half</a>															

### Bits [31:0]

[MPIDR\\_EL1](#) high half. Read-only copy of the high half of [MPIDR\\_EL1](#), as seen from the highest implemented Exception level.

## Accessing the EDDEVAFF1

EDDEVAFF1 can be accessed through the external debug interface:

Component	Offset
Debug	0xFAC

# EDDEVARCH, External Debug Device Architecture register

The EDDEVARCH characteristics are:

## Purpose

Identifies the programmers' model architecture of the external debug component.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDDEVARCH is in the Debug power domain.

Implementation of this register is OPTIONAL.

## Attributes

EDDEVARCH is a 32-bit register.

## Field descriptions

The EDDEVARCH 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
ARCHITECT											PRESENT	REVISION				ARCHID															

### ARCHITECT, bits [31:21]

Defines the architecture of the component. For debug, this is ARM Limited.

Bits [31:28] are the JEP106 continuation code, 0×4.

Bits [27:21] are the JEP106 ID code, 0×3B.

### PRESENT, bit [20]

When set to 1, indicates that the DEVARCH is present.

This field is 1 in ARMv8.

### REVISION, bits [19:16]

Defines the architecture revision. For architectures defined by ARM this is the minor revision.

For debug, the revision defined by ARMv8 is 0×0.

All other values are reserved.

**ARCHID, bits [15:0]**

Defines this part to be an ARMv8 debug component. For architectures defined by ARM this is further subdivided.

For debug:

- Bits [15:12] are the architecture version.
  - In an ARMv8.0 implementation, the only permitted value is 0x6.
  - In an ARMv8.1 implementation that includes ARMv8.1-VHE, the only permitted value is 0x7.
  - In an ARMv8.1 implementation that does not include ARMv8.1-VHE, the permitted values are 0x6 and 0x7.
  - In an ARMv8.2 implementation, the only permitted value is 0x8.
- Bits [11:0] are the architecture part number, 0xA15.

This corresponds to the ARMv8 debug architecture version.

**Accessing the EDDEVARCH**

EDDEVARCH can be accessed through the external debug interface:

Component	Offset
Debug	0xFBC

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDDEVID, External Debug Device ID register 0

The EDDEVID characteristics are:

## Purpose

Provides extra information for external debuggers about features of the debug implementation.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDDEVID is in the Debug power domain.

## Attributes

EDDEVID is a 32-bit register.

## Field descriptions

The EDDEVID 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	AuxRegs				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	PCSample			

### Bits [31:28]

Reserved, RES0.

### AuxRegs, bits [27:24]

Indicates support for Auxiliary registers. Permitted values for this field are:

AuxRegs	Meaning
0000	None supported.
0001	Support for External Debug Auxiliary Control Register, <a href="#">EDACR</a> .

All other values are reserved.

### Bits [23:4]

Reserved, RES0.

### PCSample, bits [3:0]

Indicates the level of PC Sample-based Profiling support using external debug registers. Permitted values of this field are:

PCSample	Meaning
0000	Architecture-defined form of PC Sample-based Profiling not implemented using external debug registers.
0010	Only <a href="#">EDPCSR</a> and <a href="#">EDCIDSR</a> are implemented. This option is only permitted if EL3 and EL2 are not implemented.
0011	<a href="#">EDPCSR</a> , <a href="#">EDCIDSR</a> , and <a href="#">EDVIDSR</a> are implemented.

All other values are reserved.

From ARMv8.2 onwards, the only permitted value is 0b0000. The architecture defines the functionality in a different set of registers, see [PMDEVID](#).

## Accessing the EDDEVID

EDDEVID can be accessed through the external debug interface:

Component	Offset
Debug	0xFC8

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDDEVID1, External Debug Device ID register 1

The EDDEVID1 characteristics are:

## Purpose

Provides extra information for external debuggers about features of the debug implementation.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDDEVID1 is in the Debug power domain.

## Attributes

EDDEVID1 is a 32-bit register.

## Field descriptions

The EDDEVID1 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	0
																															PCSROffset

### Bits [31:4]

Reserved, RES0.

### PCSROffset, bits [3:0]

This field indicates the offset applied to PC samples returned by reads of [EDPCSR](#). Permitted values of this field in ARMv8 are:

PCSROffset	Meaning
0000	<a href="#">EDPCSR</a> not implemented.
0010	<a href="#">EDPCSR</a> implemented, and samples have no offset applied and do not sample the instruction set state in AArch32 state.

From ARMv8.2 onwards, the only permitted value is 0b0000. The architecture defines the functionality in a different set of registers, see [PMDEVID](#).

## Accessing the EDDEVID1

EDDEVID1 can be accessed through the external debug interface:

Component	Offset
Debug	0xFC4





# EDDEVID2, External Debug Device ID register 2

The EDDEVID2 characteristics are:

## Purpose

Reserved for future descriptions of features of the debug implementation.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDDEVID2 is in the Debug power domain.

## Attributes

EDDEVID2 is a 32-bit register.

## Field descriptions

The EDDEVID2 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	0

### Bits [31:0]

Reserved, RES0.

## Accessing the EDDEVID2

EDDEVID2 can be accessed through the external debug interface:

Component	Offset
Debug	0xFC0

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDDEVTYPE, External Debug Device Type register

The EDDEVTYPE characteristics are:

## Purpose

Indicates to a debugger that this component is part of a PE's debug logic.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDDEVTYPE is in the Debug power domain.

Implementation of this register is OPTIONAL.

## Attributes

EDDEVTYPE is a 32-bit register.

## Field descriptions

The EDDEVTYPE 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	SUB				MAJOR			

### Bits [31:8]

Reserved, RES0.

### SUB, bits [7:4]

Subtype. Must read as 0x1 to indicate this is a component within a PE.

### MAJOR, bits [3:0]

Major type. Must read as 0x5 to indicate this is a debug logic component.

## Accessing the EDDEVTYPE

EDDEVTYPE can be accessed through the external debug interface:

Component	Offset
Debug	0xFCC



# EDDFR, External Debug Feature Register

The EDDFR characteristics are:

## Purpose

Provides top level information about the debug system.

### Note

Debuggers must use [EDDEVARCH](#) to determine the Debug architecture version.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers' in the ARMv8 ARM, section D7.1.3.

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

It is IMPLEMENTATION DEFINED whether EDDFR is implemented in the Core power domain or in the Debug power domain.

## Attributes

EDDFR is a 64-bit register.

## Field descriptions

The EDDFR 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
CTX_CMPs				0	0	0	0	WRPs				0	0	0	0	BRPs				PMUVer				TraceVer				UNKNOWN			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:32]

Reserved, RES0.

### CTX\_CMPs, bits [31:28]

Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints.

In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of [ID\\_AA64DFR0\\_EL1](#).CTX\_CMPs.

### Bits [27:24]

Reserved, RES0.

**WRPs, bits [23:20]**

Number of watchpoints, minus 1. The value of 0b0000 is reserved.

In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of [ID\\_AA64DFR0\\_EL1](#).WRPs.

**Bits [19:16]**

Reserved, RES0.

**BRPs, bits [15:12]**

Number of breakpoints, minus 1. The value of 0b0000 is reserved.

In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of [ID\\_AA64DFR0\\_EL1](#).BRPs.

**PMUVer, bits [11:8]**

Performance Monitors Extension version. Indicates whether System register interface to Performance Monitors extension is implemented. Defined values are:

PMUVer	Meaning
0000	Performance Monitors Extension System registers not implemented.
0001	Performance Monitors Extension System registers implemented, PMUv3.
0100	Performance Monitors Extension System registers implemented, PMUv3, with a 16-bit evtCount field, and if EL2 is implemented, the addition of the MDCR_EL2.HPMD bit.
1111	IMPLEMENTATION DEFINED form of performance monitors supported, PMUv3 not supported.

All other values are reserved.

In ARMv8.0 the permitted values are 0b0000, 0b0001 and 0b1111.

From ARMv8.1 the permitted values are 0b0000, 0b0100 and 0b1111.

In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of [ID\\_AA64DFR0\\_EL1](#).PMUVer.

**TraceVer, bits [7:4]**

Trace support. Indicates whether System register interface to a trace macrocell is implemented. Defined values are:

TraceVer	Meaning
0000	Trace macrocell System registers not implemented.
0001	Trace macrocell System registers implemented.

All other values are reserved.

A value of 0b0000 only indicates that no System register interface to a trace macrocell is implemented. A trace macrocell might nevertheless be implemented without a System register interface.

In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of [ID\\_AA64DFR0\\_EL1](#).TraceVer.

**UNKNOWN, bits [3:0]**

Reserved, UNKNOWN.

## Accessing the EDDFR

EDDFR[31:0] can be accessed through the external debug interface:

Component	Offset
Debug	0xD28

EDDFR[63:32] can be accessed through the external debug interface:

Component	Offset
Debug	0xD2C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDECCR, External Debug Exception Catch Control Register

The EDECCR characteristics are:

## Purpose

Controls Exception Catch debug events.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
Error	Error	Error	RO	RW

## Configuration

External register EDECCR is architecturally mapped to AArch64 System register [OSECCR\\_EL1](#).

External register EDECCR is architecturally mapped to AArch32 System register [DBGOSECCR](#).

EDECCR is in the Core power domain. Some or all RW fields of this register have defined reset 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.

## Attributes

EDECCR is a 32-bit register.

## Field descriptions

The EDECCR 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	NSR<n>				SR<n>			NSE<n>			SE<n>					

### Bits [31:16]

Reserved, RES0.

### NSR<n>, bits [15:12]

In ARMv8.2:

Controls Non-secure exception catch on exception return to EL<n> in conjunction with NSE<n>. If EL3 and EL2 are not implemented and the PE behaves as if [SCR\\_EL3](#).NS is set to 0, this field is reserved, RES0. Otherwise, possible values for this field are:

NSR<n>	Meaning
0	<p>If the corresponding NSE&lt;n&gt; bit is 0, then Exception Catch debug events are disabled for Non-secure Exception level &lt;n&gt;.</p> <p>If the corresponding NSE&lt;n&gt; bit is 1, then Exception Catch debug events are enabled for exception entry, reset entry and exception return to Non-secure Exception level &lt;n&gt;.</p>
1	<p>If the corresponding NSE&lt;n&gt; bit is 0, then Exception Catch debug events are enabled for exception returns to Non-secure Exception level &lt;n&gt;.</p> <p>If the corresponding NSE&lt;n&gt; bit is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Non-secure Exception level &lt;n&gt;.</p>



**Note**

It is IMPLEMENTATION DEFINED whether a reset entry to an Exception level is permitted to generate an Exception Catch debug event.

NSR[3] is RES0.

If EL2 is not implemented, NSR[2] is RES0.

A value that enables an Exception Catch debug event for an Exception level that is not implemented is reserved. If this field is programmed with a reserved value then:

- The PE behaves as if it is programmed with a defined value, other than for a read of EDECCR.
- The value returned for NSR<n> by a read of EDECCR is UNKNOWN.

When this register has an architecturally-defined reset value, this field resets to 0.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**SR<n>, bits [11:8]****In ARMv8.2:**

Controls Secure exception catch on exception return to EL<n> in conjunction with SE<n>. If EL3 is not implemented and the PE behaves as if [SCR\\_EL3.NS](#) is set to 1, this field is reserved, RES0. Otherwise, possible values for this field are:

SR<n>	Meaning
0	<p>If the corresponding SE&lt;n&gt; bit is 0, then Exception Catch debug events are disabled for Secure Exception level &lt;n&gt;.</p> <p>If the corresponding SE&lt;n&gt; bit is 1, then Exception Catch debug events are enabled for exception entry, reset entry and exception return to Secure Exception level &lt;n&gt;.</p>
1	<p>If the corresponding SE&lt;n&gt; bit is 0, then Exception Catch debug events are enabled for exception returns to Secure Exception level &lt;n&gt;.</p> <p>If the corresponding SE&lt;n&gt; bit is 1, then Exception Catch debug events are enabled for exception entry and reset entry to Secure Exception level &lt;n&gt;.</p>

**Note**

It is IMPLEMENTATION DEFINED whether a reset entry to an Exception level is permitted to generate an Exception Catch debug event.

SR[2] is RES0.

If EL3 is not implemented, SR[3] is RES0.

A value that enables an Exception Catch debug event for an Exception level that is not implemented is reserved. If this field is programmed with a reserved value then:

- The PE behaves as if it is programmed with a defined value, other than for a read of EDECCR.
- The value returned for SR<n> by a read of EDECCR is UNKNOWN.

If ExternalSecureInvasiveDebugEnabled() == FALSE, then this field is ignored.

When this register has an architecturally-defined reset value, this field resets to 0.

**In ARMv8.1 and ARMv8.0:**

Reserved, RES0.

**NSE<n>, bits [7:4]****In ARMv8.2:**

Coarse-grained Non-secure exception catch for EL<n>. This controls whether Exception Catch debug events are enabled for Non-secure EL<n>. This also controls:

- The behavior of exception catch on exception entry to EL<n>.
- The behavior of exception catch on exception return to EL<n> in conjunction with NSR<n>.

If EL3 and EL2 are not implemented and the PE behaves as if [SCR\\_EL3.NS](#) is set to 0, this field is reserved, RES0. Otherwise, possible values for this field are:

NSE<n>	Meaning
0	Exception Catch debug events are disabled for Non-secure Exception level <n>.
1	Exception Catch debug events are enabled for Non-secure Exception level <n>.

NSE[3,0] are RES0.

A value that enables an Exception Catch debug event for an Exception level that is not implemented is reserved. If this field is programmed with a reserved value then:

- The PE behaves as if it is programmed with a defined value, other than for a read of EDECCR.
- The value returned for NSE<n> by a read of EDECCR is UNKNOWN.

When this register has an architecturally-defined reset value, this field resets to 0.

**In ARMv8.1 and ARMv8.0:**

Coarse-grained Non-secure exception catch. If EL3 and EL2 are not implemented and the PE behaves as if [SCR\\_EL3.NS](#) is set to 0, this field is reserved, RES0. Otherwise, possible values for this field are:

NSE	Meaning
0000	Exception Catch debug event disabled for Non-secure Exception levels.
0010	Exception Catch debug event enabled for Non-secure EL1.
0100	Exception Catch debug event enabled for Non-secure EL2.
0110	Exception Catch debug event enabled for Non-secure EL1 and EL2.

All other values are reserved. A value that enables an Exception Catch debug event for an Exception level that is not implemented is reserved. If this field is programmed with a reserved value then:

- The PE behaves as if it is programmed with a defined value, other than for a read of EDECCR.
- The value returned for NSE by a read of EDECCR is UNKNOWN.

**SE<n>, bits [3:0]****In ARMv8.2:**

Coarse-grained Secure exception catch for EL<n>. This field controls whether Exception Catch debug events are enabled for Secure EL<n>.

- The behavior of exception catch on exception entry to EL<n>.
- The behavior of exception catch on exception return to EL<n> in conjunction with SR<n>.

If EL3 is not implemented and the PE behaves as if [SCR\\_EL3.NS](#) is set to 1, this field is reserved, RES0. Otherwise, possible values for this field are:

SE<n>	Meaning
0	Exception Catch debug events are disabled for Secure Exception level <n>.
1	Exception Catch debug events are enabled for Secure Exception level <n>.

SE[2,0] are RES0.

If EL3 is not implemented, SE[3] is RES0.

A value that enables an Exception Catch debug event for an Exception level that is not implemented is reserved. If this field is programmed with a reserved value then:

- The PE behaves as if it is programmed with a defined value, other than for a read of EDECCR.
- The value returned for SE<n> by a read of EDECCR is UNKNOWN.

If `ExternalSecureInvasiveDebugEnabled() == FALSE`, then this field is ignored.

When this register has an architecturally-defined reset value, this field resets to 0.

#### In ARMv8.1 and ARMv8.0:

Coarse-grained Secure exception catch. If EL3 is not implemented and the PE behaves as if [SCR\\_EL3.NS](#) is set to 1, this field is reserved, RES0. Otherwise, possible values for this field are:

SE	Meaning
0000	Exception Catch debug event disabled for Secure Exception levels.
0010	Exception Catch debug event enabled for Secure EL1.
1000	Exception Catch debug event enabled for Secure EL3.
1010	Exception Catch debug event enabled for Secure EL1 and EL3.

All other values are reserved. A value that enables an Exception Catch debug event for an Exception level that is not implemented is reserved. If this field is programmed with a reserved value then:

- The PE behaves as if it is programmed with a defined value, other than for a read of EDECCR.
- The value returned for SE by a read of EDECCR is UNKNOWN.

The `NSR<n>`, `SR<n>`, `NSE<n>`, and `SE<n>` fields combine to control Exception Catch as follows:

(N)SR<n>	(N)SE<n>	Behavior on exception return to ELn	Behavior on exception taken to ELn
0	0	No action	No action
0	1	Halt if allowed	Halt if allowed
1	0	Halt if allowed	No action
1	1	No action	Halt if allowed

## Accessing the EDECCR

EDECCR can be accessed through the external debug interface:

Component	Offset
Debug	0x098

# EDECR, External Debug Execution Control Register

The EDECR characteristics are:

## Purpose

Controls Halting debug events.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RW

## Configuration

EDECR is in the Debug power domain. Some or all RW fields of this register have defined reset values. These apply only on an External debug reset. The register is not affected by a Warm reset and is not affected by a Cold reset.

## Attributes

EDECR is a 32-bit register.

## Field descriptions

The EDECR 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	SS	RCE	OSUCE

### Bits [31:3]

Reserved, RES0.

### SS, bit [2]

Halting step enable. Possible values of this field are:

SS	Meaning
0	Halting step debug event disabled.
1	Halting step debug event enabled.

If the value of EDECR.SS is changed when the PE is in Non-debug state, behavior is CONSTRAINED UNPREDICTABLE as described in 'Changing the value of EDECR.SS when not in Debug state' in the ARM ARM, section H3.2.5.

When this register has an architecturally-defined reset value, this field resets to 0.

### RCE, bit [1]

Reset Catch enable. Possible values of this field are:

RCE	Meaning
0	Reset Catch debug event disabled.
1	Reset Catch debug event enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

### OSUCE, bit [0]

OS Unlock Catch enabled. Possible values of this field are:

OSUCE	Meaning
0	OS Unlock Catch debug event disabled.
1	OS Unlock Catch debug event enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the EDECR

EDECR can be accessed through the external debug interface:

Component	Offset
Debug	0x024

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDES, External Debug Event Status Register

The EDES characteristics are:

## Purpose

Indicates the status of internally pending Halting debug events.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	SLK	Default
Error	Error	RO	RW

If a request to clear a pending Halting debug event is received at or about the time when halting becomes allowed, it is **CONSTRAINED UNPREDICTABLE** whether the event is taken.

If Core power is removed while a Halting debug event is pending, it is lost. However, it might become pending again when the Core is powered back on and Cold reset.

## Configuration

EDES is in the Core power domain. Some or all RW fields of this register have defined reset values. These apply on a Warm or Cold reset. The register is not affected by an External debug reset.

## Attributes

EDES is a 32-bit register.

## Field descriptions

The EDES 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	SS	RC	OSUC

### Bits [31:3]

Reserved, RES0.

### SS, bit [2]

Halting step debug event pending. Possible values of this field are:

SS	Meaning
0	Reading this means that a Halting step debug event is not pending. Writing this means no action.
1	Reading this means that a Halting step debug event is pending. Writing this clears the pending Halting step debug event.

When this register has an architecturally-defined reset value, this field resets to the value of [EDEC.R.SS](#).

**RC, bit [1]**

Reset Catch debug event pending. Possible values of this field are:

RC	Meaning
0	Reading this means that a Reset Catch debug event is not pending. Writing this means no action.
1	Reading this means that a Reset Catch debug event is pending. Writing this clears the pending Reset Catch debug event.

When this register has an architecturally-defined reset value, this field resets to the value of [EDECR](#).RCE.

**OSUC, bit [0]**

OS Unlock Catch debug event pending. Possible values of this field are:

OSUC	Meaning
0	Reading this means that an OS Unlock Catch debug event is not pending. Writing this means no action.
1	Reading this means that an OS Unlock Catch debug event is pending. Writing this clears the pending OS Unlock Catch debug event.

When this register has an architecturally-defined reset value, this field resets to 0.

**Accessing the EDES**

EDES can be accessed through the external debug interface:

Component	Offset
Debug	0x020

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDITCTRL, External Debug Integration mode Control register

The EDITCTRL characteristics are:

## Purpose

Enables the external debug to switch from its default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	Default
IMP DEF	IMP DEF	IMP DEF	RW

## Configuration

It is IMPLEMENTATION DEFINED whether EDITCTRL is implemented in the Core power domain or in the Debug power domain. Some or all RW fields of this register have defined reset values, and:

- The register is not affected by a Warm reset.
- If the register is implemented in the Core power domain the reset values apply on a Cold reset, and the register is not affected by an External debug reset.
- If the register is implemented in the Debug power domain the reset values apply on an External debug reset, and the register is not affected by a Cold reset.

Implementation of this register is OPTIONAL.

## Attributes

EDITCTRL is a 32-bit register.

## Field descriptions

The EDITCTRL 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	IME

### Bits [31:1]

Reserved, RES0.

### IME, bit [0]

Integration mode enable. When IME == 1, the device reverts to an integration mode to enable integration testing or topology detection. The integration mode behavior is IMPLEMENTATION DEFINED.

IME	Meaning
0	Normal operation.
1	Integration mode enabled.

When this register has an architecturally-defined reset value, this field resets to 0.



## Accessing the EDITCTRL

EDITCTRL can be accessed through the external debug interface:

Component	Offset
Debug	0xF00

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDITR, External Debug Instruction Transfer Register

The EDITR characteristics are:

## Purpose

Used in Debug state for passing instructions to the PE for execution.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
Error	Error	Error	WI	WO

If [EDSCR.ITE](#) == 0 when the PE exits Debug state on receiving a Restart request trigger event, the behavior of any instruction issued through the ITR in Normal access mode that has not completed execution is CONSTRAINED UNPREDICTABLE, and must do one of the following:

- It must complete execution in Debug state before the PE executes the restart sequence.
- It must complete execution in Non-debug state before the PE executes the restart sequence.
- It must be abandoned. This means that the instruction does not execute. Any registers or memory accessed by the instruction are left in an UNKNOWN state.

EDITR ignores writes if the PE is in Non-debug state.

## Configuration

EDITR is in the Core power domain.

## Attributes

EDITR is a 32-bit register.

## Field descriptions

The EDITR bit assignments are:

### When in AArch32 state:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<a href="#">T32Second</a>																<a href="#">T32First</a>															

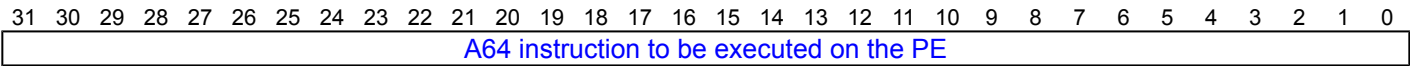
#### T32Second, bits [31:16]

Second halfword of the T32 instruction to be executed on the PE. When EDITR contains a 16-bit T32 instruction, this field is ignored. For more information see 'Behavior in Debug state' in the ARMv8 ARM, section H2, Debug State.

#### T32First, bits [15:0]

First halfword of the T32 instruction to be executed on the PE.

When in AArch64 state:



Bits [31:0]

A64 instruction to be executed on the PE.

Accessing the EDITR

EDITR can be accessed through the external debug interface:

Component	Offset
Debug	0x084

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDLAR, External Debug Lock Access Register

The EDLAR characteristics are:

## Purpose

Allows or disallows access to the external debug registers through a memory-mapped interface.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
WO

## Configuration

EDLAR is in the Debug power domain.

If OPTIONAL memory-mapped access to the external debug interface is supported then an OPTIONAL Software Lock can be implemented as part of CoreSight compliance.

EDLAR ignores writes if the Software Lock is not implemented and ignores writes for other accesses to the external debug interface.

The Software Lock provides a lock to prevent memory-mapped writes to the debug registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the debug registers. It does not, and cannot, prevent all accidental or malicious damage.

Software uses EDLAR to set or clear the lock, and [EDLSR](#) to check the current status of the lock.

## Attributes

EDLAR is a 32-bit register.

## Field descriptions

The EDLAR 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
																KEY															

### KEY, bits [31:0]

Lock Access control. Writing the key value 0xC5ACCE55 to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface.

Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.

## Accessing the EDLAR

EDLAR can be accessed through a memory-mapped access to the external debug interface:

Component	Offset
Debug	0xFB0



# EDLSR, External Debug Lock Status Register

The EDLSR characteristics are:

## Purpose

Indicates the current status of the software lock for external debug registers.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDLSR is in the Debug power domain. Some or all RW fields of this register have defined reset values. These apply only on an External debug reset. The register is not affected by a Warm reset and is not affected by a Cold reset.

If OPTIONAL memory-mapped access to the external debug interface is supported then an OPTIONAL Software Lock can be implemented as part of CoreSight compliance.

EDLSR is RAZ if the Software Lock is not implemented and is RAZ for other accesses to the external debug interface.

The Software Lock provides a lock to prevent memory-mapped writes to the debug registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the debug registers. It does not, and cannot, prevent all accidental or malicious damage.

Software uses [EDLAR](#) to set or clear the lock, and EDLSR to check the current status of the lock.

## Attributes

EDLSR is a 32-bit register.

## Field descriptions

The EDLSR 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	<a href="#">nTT</a>	<a href="#">SLK</a>	<a href="#">SLI</a>

### Bits [31:3]

Reserved, RES0.

### nTT, bit [2]

Not thirty-two bit access required. RAZ.

### SLK, bit [1]

Software Lock status for this component. For an access to LSR that is not a memory-mapped access, or when the Software Lock is not implemented, this field is RES0.

For memory-mapped accesses when the Software Lock is implemented, possible values of this field are:

SLK	Meaning
0	Lock clear. Writes are permitted to this component's registers.
1	Lock set. Writes to this component's registers are ignored, and reads have no side effects.

When this register has an architecturally-defined reset value, this field resets to 1.

### SLI, bit [0]

Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is IMPLEMENTATION DEFINED. Permitted values are:

SLI	Meaning
0	Software Lock not implemented or not memory-mapped access.
1	Software Lock implemented and memory-mapped access.

## Accessing the EDLSR

EDLSR can be accessed through a memory-mapped access to the external debug interface:

Component	Offset
Debug	0xFB4

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDPCSR, External Debug Program Counter Sample Register

The EDPCSR characteristics are:

## Purpose

Holds a sampled instruction address value.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
Error	Error	Error	RO	RO

## Configuration

EDPCSR 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 if the OPTIONAL PC Sample-based Profiling Extension is implemented and ARMv8.2-PCSample is not implemented. If ARMv8.2-PCSample is implemented, this register is RES0 and the architecture defines the functionality in [PMPCSR](#).

## Attributes

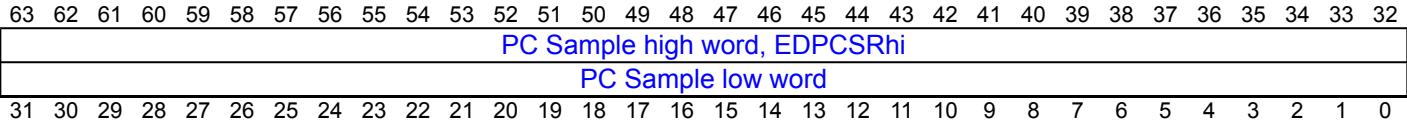
EDPCSR is a pair of 32-bit registers.

If ARMv8.1-VHE is implemented, the format of this register differs depending on the value of [EDSCR](#).SC2.

## Field descriptions

The EDPCSR bit assignments are:

### When ARMv8.1-VHE is not implemented:



#### Bits [63:32]

PC Sample high word, EDPCSRhi. If [EDVIDSR](#).HV == 0 then this field is RAZ, otherwise bits [63:32] of the sampled instruction address value. The translation regime that EDPCSR samples can be determined from [EDVIDSR](#).{NS,E2,E3}.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Bits [31:0]

PC Sample low word. EDPCSRlo, bits[31:0] of the sampled instruction address value. The translation regime that EDPCSR samples can be determined from [EDVIDSR](#).{NS,E2,E3}.



- For a read of EDPCSR[31:0] from the memory-mapped interface, if EDLSR.SLK == 1, meaning the OPTIONAL Software Lock is locked, then the access has no side-effects.
- In any other cases, a read of EDPCSR[31:0] has the side-effect of indirectly writing to EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#):
  - If the PE is in Debug state, or PC Sample-based profiling is prohibited, EDPCSRlo reads as 0xFFFFFFFF, and EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#) become UNKNOWN.
  - If the PE is in Reset state, the sampled value is UNKNOWN and EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#) 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 EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#) become UNKNOWN.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### When ARMv8.1-VHE is implemented and EDSCR.SC2 == 0:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
PC Sample high word, EDPCSRhi																															
PC Sample low word																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

#### Bits [63:32]

PC Sample high word, EDPCSRhi. If [EDVIDSR](#).HV == 0 then this field is RAZ, otherwise bits [63:32] of the sampled instruction address value. The translation regime that EDPCSR samples can be determined from [EDVIDSR](#).{NS,E2,E3}.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Bits [31:0]

PC Sample low word. EDPCSRlo, bits[31:0] of the sampled instruction address value. The translation regime that EDPCSR samples can be determined from [EDVIDSR](#).{NS,E2,E3}.

- For a read of EDPCSR[31:0] from the memory-mapped interface, if EDLSR.SLK == 1, meaning the OPTIONAL Software Lock is locked, then the access has no side-effects.
- In any other cases, a read of EDPCSR[31:0] has the side-effect of indirectly writing to EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#):
  - If the PE is in Debug state, or PC Sample-based profiling is prohibited, EDPCSRlo reads as 0xFFFFFFFF, and EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#) become UNKNOWN.
  - If the PE is in Reset state, the sampled value is UNKNOWN and EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#) 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 EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#) become UNKNOWN.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### When ARMv8.1-VHE is implemented and EDSCR.SC2 == 1:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
NS	EL	0	0	0	0	0	PC Sample high word, EDPCSRhi																								
PC Sample low word																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

#### NS, bit [63]

Non-secure state sample. Indicates the Security state that is associated with the most recent EDPCSR sample or, when it is read as a single atomic 64-bit read, the current EDPCSR sample. The translation regime that EDPCSR samples can be determined from EDPCSR.{NS,EL}.

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 EDPCSR sample or, when it is read as a single atomic 64-bit read, the current EDPCSR sample. The translation regime that EDPCSR samples can be determined from EDPCSR.{NS,EL}.

EL	Meaning
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.

#### Bits [55:32]

PC Sample high word, EDPCSRhi. Bits [55:32] of the sampled instruction address value. The translation regime that EDPCSR samples can be determined from EDPCSR.{NS,EL}.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Bits [31:0]

PC Sample low word. EDPCSRlo, bits[31:0] of the sampled instruction address value. The translation regime that EDPCSR samples can be determined from [EDVIDSR](#).{NS,E2,E3}.

- For a read of EDPCSR[31:0] from the memory-mapped interface, if EDLSR.SLK = 1, meaning the OPTIONAL Software Lock is locked, then the access has no side-effects.
- In any other cases, a read of EDPCSR[31:0] has the side-effect of indirectly writing to EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#):
  - If the PE is in Debug state, or PC Sample-based profiling is prohibited, EDPCSRlo reads as 0xFFFFFFFF, and EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#) become UNKNOWN.
  - If the PE is in Reset state, the sampled value is UNKNOWN and EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#) 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 EDPCSRhi, [EDCIDSR](#), and [EDVIDSR](#) become UNKNOWN.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### When ARMv8.2-PCSample is implemented:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
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	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 [63:0]

Reserved, RES0.

## Accessing the EDPCSR

EDPCSR[31:0] can be accessed through its memory-mapped interface:

Component	Offset
Debug	0x0A0

EDPCSR[63:32] can be accessed through its memory-mapped interface:

Component	Offset
Debug	0x0AC



# EDPFR, External Debug Processor Feature Register

The EDPFR characteristics are:

## Purpose

Provides information about implemented PE features.

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.

## Usage constraints

This register is accessible as follows:

Off	DLK	Default
IMP DEF	IMP DEF	RO

## Configuration

It is IMPLEMENTATION DEFINED whether EDPFR is implemented in the Core power domain or in the Debug power domain.

## Attributes

EDPFR is a 64-bit register.

## Field descriptions

The EDPFR 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				
UNK				GIC				AdvSIMD				FP				EL3				EL2				EL1				EL0			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:36]

Reserved, RES0.

### SVE, bits [35:32]

In 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.

**UNK, bits [31:28]**

When the RAS Extension is implemented, this field is UNKNOWN. Otherwise, this field is RES0.

**Note**

ARMv8.2 requires the implementation of the RAS Extension.

**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.

In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of [ID\\_AA64PFR0\\_EL1](#).GIC.

**AdvSIMD, bits [23:20]**

Advanced SIMD. Defined values are:

AdvSIMD	Meaning
0000	Advanced SIMD is implemented, including support for the following SISO and SIMD operations: <ul style="list-style-type: none"> <li>Integer byte, halfword, word and doubleword element operations.</li> <li>Single-precision and double-precision floating-point arithmetic.</li> <li>Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.</li> </ul>
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.

In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of [ID\\_AA64PFR0\\_EL1](#).AdvSIMD.

**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"> <li>Single-precision and double-precision floating-point types.</li> <li>Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.</li> </ul>
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, that includes the ARMv8.2-FP16 extension.
- 1111 in an implementation without floating-point support.

In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of [ID\\_AA64PFR0\\_EL1.FP](#).

### EL3, bits [15:12]

AArch64 EL3 Exception level handling. Defined values are:

EL3	Meaning
0000	EL3 is not implemented or cannot be executed in AArch64 state.
0001	EL3 can be executed in AArch64 state only.
0010	EL3 can be executed in either AArch64 or AArch32 state.

When the value of [EDAA32PFR.EL3](#) is non-zero, this field must be 0000.

All other values are reserved.

In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of [ID\\_AA64PFR0\\_EL1.EL3](#).

### EL2, bits [11:8]

AArch64 EL2 Exception level handling. Defined values are:

EL2	Meaning
0000	EL2 is not implemented or cannot be executed in AArch64 state.
0001	EL2 can be executed in AArch64 state only.
0010	EL2 can be executed in either AArch64 or AArch32 state.

When the value of [EDAA32PFR.EL2](#) is non-zero, this field must be 0000.

All other values are reserved.

In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of [ID\\_AA64PFR0\\_EL1.EL2](#).

### EL1, bits [7:4]

AArch64 EL1 Exception level handling. Defined values are:

EL1	Meaning
0000	EL1 can be executed in AArch32 state only.
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.

In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of [ID\\_AA64PFR0\\_EL1.EL1](#).

### EL0, bits [3:0]

AArch64 EL0 Exception level handling. Defined values are:

EL0	Meaning
0000	EL0 can be executed in AArch32 state only.
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.

In an ARMv8-A implementation that supports AArch64 state in at least one Exception level, this field returns the value of [ID\\_AA64PFR0\\_EL1](#).EL0.

## Accessing the EDPFR

EDPFR[31:0] can be accessed through the external debug interface:

Component	Offset
Debug	0xD20

EDPFR[63:32] can be accessed through the external debug interface:

Component	Offset
Debug	0xD24

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDPIDR0, External Debug Peripheral Identification Register 0

The EDPIDR0 characteristics are:

## Purpose

Provides information to identify an external debug component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDPIDR0 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

EDPIDR0 is a 32-bit register.

## Field descriptions

The EDPIDR0 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								

[PART\\_0](#)

### Bits [31:8]

Reserved, RES0.

### PART\_0, bits [7:0]

Part number, least significant byte.

## Accessing the EDPIDR0

EDPIDR0 can be accessed through the external debug interface:

Component	Offset
Debug	0xFE0





# EDPIDR1, External Debug Peripheral Identification Register 1

The EDPIDR1 characteristics are:

## Purpose

Provides information to identify an external debug component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDPIDR1 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

EDPIDR1 is a 32-bit register.

## Field descriptions

The EDPIDR1 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	DES_0				PART_1			

### Bits [31:8]

Reserved, RES0.

### DES\_0, bits [7:4]

Designer, least significant nibble of JEP106 ID code. For ARM Limited, this field is 0b1011.

### PART\_1, bits [3:0]

Part number, most significant nibble.

## Accessing the EDPIDR1

EDPIDR1 can be accessed through the external debug interface:

Component	Offset
-----------	--------

Debug	0xFE4
-------	-------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDPIDR2, External Debug Peripheral Identification Register 2

The EDPIDR2 characteristics are:

## Purpose

Provides information to identify an external debug component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDPIDR2 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

EDPIDR2 is a 32-bit register.

## Field descriptions

The EDPIDR2 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	REVISION	JEDEC	DES	1				

### Bits [31:8]

Reserved, RES0.

### REVISION, bits [7:4]

Part major revision. Parts can also use this field to extend Part number to 16-bits.

### JEDEC, bit [3]

RAO. Indicates a JEP106 identity code is used.

### DES\_1, bits [2:0]

Designer, most significant bits of JEP106 ID code. For ARM Limited, this field is 0b011.

## Accessing the EDPIDR2

EDPIDR2 can be accessed through the external debug interface:

Component	Offset
Debug	0xFE8

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDPIDR3, External Debug Peripheral Identification Register 3

The EDPIDR3 characteristics are:

## Purpose

Provides information to identify an external debug component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDPIDR3 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

EDPIDR3 is a 32-bit register.

## Field descriptions

The EDPIDR3 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	7	6	5	4	3	2	1	0
																								REVAND				CMOD			

### Bits [31:8]

Reserved, RES0.

### REVAND, bits [7:4]

Part minor revision. Parts using [EDPIDR2](#).REVISION as an extension to the Part number must use this field as a major revision number.

### CMOD, bits [3:0]

Customer modified. Indicates someone other than the Designer has modified the component.

## Accessing the EDPIDR3

EDPIDR3 can be accessed through the external debug interface:

Component	Offset
-----------	--------

Debug	0xFEC
-------	-------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDPIDR4, External Debug Peripheral Identification Register 4

The EDPIDR4 characteristics are:

## Purpose

Provides information to identify an external debug component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

EDPIDR4 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

EDPIDR4 is a 32-bit register.

## Field descriptions

The EDPIDR4 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	SIZE				DES 2			

### Bits [31:8]

Reserved, RES0.

### SIZE, bits [7:4]

Size of the component. RAZ. Log2 of the number of 4KB pages from the start of the component to the end of the component ID registers.

### DES\_2, bits [3:0]

Designer, JEP106 continuation code, least significant nibble. For ARM Limited, this field is 0b0100.

## Accessing the EDPIDR4

EDPIDR4 can be accessed through the external debug interface:

Component	Offset
-----------	--------



Debug	0xFD0
-------	-------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDPRCR, External Debug Power/Reset Control Register

The EDPRCR characteristics are:

## Purpose

Controls the PE functionality related to powerup, reset, and powerdown.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RW

On permitted accesses to the register, other access controls affect the behavior of some fields. See the field descriptions for more information.

## Configuration

EDPRCR contains fields that are in the Core power domain and fields that are in the Debug power domain.

For RW fields see the field description for a description of the behavior of the field on a reset that applies to its power domain. However:

- Fields that are in the Core power domain are not affected by a warm reset and are not affected by an External debug reset.
- Fields that are in the Debug power domain reset to their defined reset values on an External debug reset, and are not affected by a Warm reset and are not affected by a Cold reset.

CORENPDRQ is the only field that is mapped between the EDPRCR and DBGPRCR and DBGPRCR\_EL1.

## Attributes

EDPRCR is a 32-bit register.

## Field descriptions

The EDPRCR 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	COREPURQ	0	CWRR	CORENPDRQ

### Bits [31:4]

Reserved, RES0.

### COREPURQ, bit [3]

Core powerup request. Allows a debugger to request that the power controller power up the core, enabling access to the debug register in the Core power domain. The actions on writing to this bit are:

COREPURQ	Meaning
0	Do not request power up of the Core power domain.
1	Request power up of the Core power domain, and emulation of powerdown.

In an implementation that includes the recommended external debug interface, this bit drives the DBGPWRUPREQ signal.

Typically, this request is passed to an external power controller. This means that whether a request causes power up is dependent on the IMPLEMENTATION DEFINED nature of the system.

This field is in the Debug power domain and can be read and written when the Core power domain is powered off. On an External debug reset this field resets to 0.

The power controller must not allow the Core power domain to switch off while this bit is 1.

When this register has an architecturally-defined reset value, this field resets to 0.

This field resets to its defined reset value on External debug reset.

This table summarizes the effect of the register access controls on the behavior of this field:

SLK	Default
RO	RW

'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.

## Bit [2]

Reserved, RES0.

## CWRR, bit [1]

Warm reset request. Write only bit that reads as zero. The actions on writing to this bit are:

CWRR	Meaning
0	No action.
1	Request Warm reset.

The PE ignores writes to this bit if any of the following are true:

- ExternalInvasiveDebugEnabled() == FALSE, EL3 is not implemented, and the implemented Security state is Non-Secure state.
- ExternalSecureInvasiveDebugEnabled() == FALSE and one of the following is true:
  - EL3 is implemented.
  - The implemented Security state is Secure state.
- The Core power domain is either off or in a low-power state where the Core power domain registers cannot be accessed.
- DoubleLockStatus() == TRUE (OS Double Lock is set).
- OSLSR.OSLK == 1 (OS lock is locked).

In an implementation that includes the recommended external debug interface, this bit drives the DBGRSTREQ signal.

The extent of the reset is IMPLEMENTATION DEFINED, but must be one of:

- The request is ignored.
- Only this PE is Warm reset.
- This PE and other components of the system, possibly including other PEs, are Warm reset.

---

### Note

Although the ARM architecture permits the first option from the above list, ARM recommends that either of the other options is implemented.

---

When this register has an architecturally-defined reset value, this field resets to 0.

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	OSLK	SLK	Default
WI	WI	WI	WI	WO

'Access permissions for the External debug interface registers' in the *ARM<sup>®</sup> 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.

## CORENPDRQ, bit [0]

Core no powerdown request. Requests emulation of powerdown. Possible values of this bit are:

CORENPDRQ	Meaning
0	If the system responds to a powerdown request, it powers down Core power domain.
1	If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain.

This bit is UNKNOWN, and the PE ignores writes to this bit if any of the following are true:

- The Core power domain is either off or in a low-power state where the Core power domain registers cannot be accessed.
- DoubleLockStatus() == TRUE (OS Double Lock is set).
- OSLSR.OSLK == 1 (OS lock is locked).

Permitted accesses to this field map to the [DBGPRCR.CORENPDRQ](#) and [DBGPRCR\\_EL1.CORENPDRQ](#) fields.

This field is in the Core reset domain. See the descriptions of the [DBGPRCR.CORENPDRQ](#) and [DBGPRCR\\_EL1.CORENPDRQ](#) fields for information about the effect of a Cold reset on the value returned by a permitted read of this field.

This table summarizes the effect of the register access controls on the behavior of this field:

Off	DLK	OSLK	SLK	Default
WI	WI	WI	RO	RW

'Access permissions for the External debug interface registers' in the *ARM<sup>®</sup> 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 EDPRCR

EDPRCR can be accessed through the external debug interface:

Component	Offset
Debug	0x310

# 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 ( $\text{EDPRSR.PU} = 1$ ), then following a read of EDPRSR:

- If  $\text{DoubleLockStatus()} = \text{FALSE}$ , then:
  - $\text{EDPRSR}\{ \text{SDR}, \text{SPMAD}, \text{SDAD}, \text{SPD} \}$  are cleared to 0.
  - $\text{EDPRSR.SR}$  is cleared to 0 if the non-debug logic of the PE is not in reset state ( $\text{EDPRSR.R} = 0$ ).
- Otherwise it is **CONSTRAINED UNPREDICTABLE** whether or not this clearing occurs.

If the Core power domain is powered down ( $\text{EDPRSR.PU} = 0$ ), then:

- $\text{EDPRSR}\{ \text{SDR}, \text{SPMAD}, \text{SDAD}, \text{SR} \}$  are all **UNKNOWN**, and are either reset or restored on being powered up.
- $\text{EDPRSR.SPD}$  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
- EDPRSR.{OSLK, R} is 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<sup>®</sup> 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
- EDPRSR.{OSLK, R} is 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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 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<sup>®</sup> 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

# EDRCR, External Debug Reserve Control Register

The EDRCR characteristics are:

## Purpose

This register is used to allow imprecise entry to Debug state and clear sticky bits in [EDSCR](#).

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
Error	Error	Error	WI	WO

## Configuration

EDRCR is in the Core power domain.

## Attributes

EDRCR is a 32-bit register.

## Field descriptions

The EDRCR 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	<a href="#">CBRRQ</a>	<a href="#">CSPA</a>	<a href="#">CSE</a>	0	0

### Bits [31:5]

Reserved, RES0.

### CBRRQ, bit [4]

Allow imprecise entry to Debug state. The actions on writing to this bit are:

CBRRQ	Meaning
0	No action.
1	Allow imprecise entry to Debug state, for example by canceling pending bus accesses.

Setting this bit to 1 allows a debugger to request imprecise entry to Debug state. An External Debug Request debug event must be pending before the debugger sets this bit to 1.

This feature is optional. If this feature is not implemented, writes to this bit are ignored.

### CSPA, bit [3]

Clear Sticky Pipeline Advance. This bit is used to clear the [EDSCR](#).PipeAdv bit to 0. The actions on writing to this bit are:

CSPA	Meaning
0	No action.
1	Clear the <a href="#">EDSCR</a> .PipeAdv bit to 0.

**CSE, bit [2]**

Clear Sticky Error. Used to clear the [EDSCR](#) cumulative error bits to 0. The actions on writing to this bit are:

CSE	Meaning
0	No action.
1	Clear the <a href="#">EDSCR</a> . {TXU, RXO, ERR} bits, and, if the PE is in Debug state, the <a href="#">EDSCR</a> .ITO bit, to 0.

**Bits [1:0]**

Reserved, RES0.

**Accessing the EDRCR**

EDRCR can be accessed through the external debug interface:

Component	Offset
Debug	0x090

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDSCR, External Debug Status and Control Register

The EDSCR characteristics are:

## Purpose

Main control register for the debug implementation.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	SLK	Default
Error	Error	Error	RO	RW

## Configuration

EDSCR is in the Core power domain. Some or all RW fields of this register have defined reset 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.

## Attributes

EDSCR is a 32-bit register.

## Field descriptions

The EDSCR 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	RXfull	TXfull	ITO	RXOTXU	PipeAdv	ITE	INTdis	TDAM	ASC2	NS	0	SDD	0	HDE		RW		EL	A	ERR											

### Bit [31]

Reserved, RES0.

### RXfull, bit [30]

DTRRX full. This bit is RO.

When this register has an architecturally-defined reset value, this field resets to 0.

### TXfull, bit [29]

DTRTX full. This bit is RO.

When this register has an architecturally-defined reset value, this field resets to 0.

### ITO, bit [28]

ITR overrun. This bit is RO.

If the PE is in Non-debug state, this bit is UNKNOWN. ITO is set to 0 on entry to Debug state.

**RXO, bit [27]**

DTRRX overrun. This bit is RO.

When this register has an architecturally-defined reset value, this field resets to 0.

**TXU, bit [26]**

DTRTX underrun. This bit is RO.

When this register has an architecturally-defined reset value, this field resets to 0.

**PipeAdv, bit [25]**

Pipeline advance. This bit is RO. Set to 1 every time the PE pipeline retires one or more instructions. Cleared to 0 by a write to [EDRCR.CSPA](#).

The architecture does not define precisely when this bit is set to 1. It requires only that this happen periodically in Non-debug state to indicate that software execution is progressing.

**ITE, bit [24]**

ITR empty. This bit is RO.

If the PE is in Non-debug state, this bit is UNKNOWN. It is always valid in Debug state.

**INTdis, bits [23:22]**

Interrupt disable. Disables taking interrupts (including virtual interrupts and System Error interrupts) in Non-Debug state.

If ExternalInvasiveDebugEnabled() = FALSE, the value of this field is ignored.

If ExternalInvasiveDebugEnabled() = TRUE, the possible values of this field are:

INTdis	Meaning
00	Do not disable interrupts.
01	Disable interrupts taken to Non-secure EL1.
10	Disable interrupts taken only to Non-secure EL1 and Non-secure EL2. If external secure invasive debug is enabled, also disable interrupts taken to Secure EL1.
11	Disable interrupts taken only to Non-secure EL1 and Non-secure EL2. If external secure invasive debug is enabled, also disable all other interrupts.

The value of INTdis does not affect whether an interrupt is a WFI wake-up event, but can mask an interrupt as a WFE wake-up event.

If EL3 and EL2 are not implemented, the values 0b01 and 0b10 are reserved. If programmed with a reserved value the PE behaves as if INTdis has been programmed with a defined value, other than for a direct read of EDSCR, and the value returned by a read of EDSCR.INTdis is UNKNOWN.

When this register has an architecturally-defined reset value, this field resets to 0.

**TDA, bit [21]**

Traps accesses to the following Debug System registers:

- AArch64: [DBGBCR<n>\\_EL1](#), [DBGBVR<n>\\_EL1](#), [DBGWCR<n>\\_EL1](#), [DBGWVR<n>\\_EL1](#).
- AArch32: [DBGBCR<n>](#), [DBGBVR<n>](#), [DBGBXVR<n>](#), [DBGWCR<n>](#), [DBGWVR<n>](#).

The possible values of this field are:

TDA	Meaning
0	Accesses to Debug System registers do not generate a Software Access debug event.
1	Accesses to Debug System registers generate a Software Access debug event, if OSLSR.OSLK is 0 and if halting is allowed.

When this register has an architecturally-defined reset value, this field resets to 0.



**MA, bit [20]**

Memory access mode. Controls use of memory-access mode for accessing ITR and the DCC. This bit is ignored if in Non-debug state and set to zero on entry to Debug state.

Possible values of this field are:

MA	Meaning
0	Normal access mode.
1	Memory access mode.

**Bit [19]**

In ARMv8.2 and ARMv8.0:

Reserved, RES0.

In ARMv8.1:

Sample [CONTEXTIDR\\_EL2](#). Controls whether the Sample-based Profiling Extension samples [CONTEXTIDR\\_EL2](#) or [VTTBR\\_EL2](#).VMID.

SC2	Meaning
0	Sample <a href="#">VTTBR_EL2</a> .VMID.
1	Sample <a href="#">CONTEXTIDR_EL2</a> .

If the PC Sample-based Profiling Extension is not implemented, then this field is RES0.

**NS, bit [18]**

Non-secure status. Read-only. When in Debug state, gives the current Security state:

NS	Meaning
0	Secure state, IsSecure() == TRUE.
1	Non-secure state, IsSecure() == FALSE.

In Non-debug state, this bit is UNKNOWN.

**Bit [17]**

Reserved, RES0.

**SDD, bit [16]**

Secure debug disabled. This bit is RO.

On entry to Debug state:

- If entering in Secure state, SDD is set to 0.
- If entering in Non-secure state, SDD is set to the inverse of ExternalSecureInvasiveDebugEnabled().

In Debug state, the value of the SDD bit does not change, even if ExternalSecureInvasiveDebugEnabled() changes.

In Non-debug state:

- SDD returns the inverse of ExternalSecureInvasiveDebugEnabled(). If the authentication signals that control ExternalSecureInvasiveDebugEnabled() change, a context synchronization event is required to guarantee their effect.
- This bit is unaffected by the Security state of the PE.

If EL3 is not implemented and the implementation is Non-secure, this bit is RES1.

**Bit [15]**

Reserved, RES0.

**HDE, bit [14]**

Halting debug enable. The possible values of this field are:

HDE	Meaning
0	Halting disabled for Breakpoint, Watchpoint and Halt Instruction debug events.
1	Halting enabled for Breakpoint, Watchpoint and Halt Instruction debug events.

When this register has an architecturally-defined reset value, this field resets to 0.

**RW, bits [13:10]**

Exception level Execution state status. Read-only. In Debug state, each bit gives the current Execution state of each EL:

RW	Meaning
1111	All Exception levels are using AArch64.
1110	EL0 is using AArch32. All other Exception levels are using AArch64.
110x	EL0 and EL1 are using AArch32. All other Exception levels are using AArch64. Never seen if EL2 is not implemented in the current Security state.
10xx	EL0, EL1, and, if implemented in the current Security state, EL2 are using AArch32. All other Exception levels are using AArch64.
0xxx	All Exception levels are using AArch32.

However:

- The value of 1110 is only permitted at EL0.
- The values 110x are not permitted if either:
  - EL2 is not implemented.
  - EL3 is implemented and SCR\_EL3.NS/SCR.NS == 0.
- The values 10xx are not permitted if EL3 is not implemented.

In Non-debug state, this field is RAO.

**EL, bits [9:8]**

Exception level. Read-only. In Debug state, this gives the current EL of the PE.

In Non-debug state, this field is RAZ.

**A, bit [7]**

System Error interrupt pending. Read-only. In Debug state, indicates whether a SError interrupt is pending:

- If [HCR\\_EL2](#).{AMO, TGE} = {1, 0} and in Non-secure EL0 or EL1, a virtual SError interrupt.
- Otherwise, a physical SError interrupt.

A	Meaning
0	No SError interrupt pending.
1	SError interrupt pending.

A debugger can read EDSCR to check whether an SError interrupt is pending without having to execute further instructions. A pending SError might indicate data from target memory is corrupted.

UNKNOWN in Non-debug state.

**ERR, bit [6]**

Cumulative error flag. This field is RO. It is set to 1 following exceptions in Debug state and on any signaled overrun or underrun on the DTR or EDITR.

When this register has an architecturally-defined reset value, this field resets to 0.

**STATUS, bits [5:0]**

Debug status flags. This field is RO.

The possible values of this field are:

STATUS	Meaning
000010	PE is in Non-debug state.
000001	PE is restarting, exiting Debug state.
000111	Breakpoint.
010011	External debug request.
011011	Halting step, normal.
011111	Halting step, exclusive.
100011	OS Unlock Catch.
100111	Reset Catch.
101011	Watchpoint.
101111	HLT instruction.
110011	Software access to debug register.
110111	Exception Catch.
111011	Halting step, no syndrome.

All other values of STATUS are reserved.

## Accessing the EDSCR

EDSCR can be accessed through the external debug interface:

Component	Offset
Debug	0x088

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDVIDSR, External Debug Virtual Context Sample Register

The EDVIDSR characteristics are:

## Purpose

Contains sampled values captured on reading [EDPCSR](#)[31:0].

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	Default
Error	Error	Error	RO

## Configuration

EDVIDSR 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 if the OPTIONAL PC Sample-based Profiling Extension is implemented and ARMv8.2-PCSample is not implemented. If ARMv8.2-PCSample is implemented, this register is RES0 and the architecture defines the functionality in [PMPCSR](#), [PMCID2SR](#), and [PMVIDSR](#).

If EL2 is not implemented and EL3 is not implemented, it is IMPLEMENTATION DEFINED whether EDVIDSR is implemented.

## Attributes

If ARMv8.1-VHE is implemented, the format of this register differs depending on the value of [EDSCR](#).SC2.

## Field descriptions

The EDVIDSR bit assignments are:

### When ARMv8.1-VHE is not 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
<a href="#">NS</a>	<a href="#">E2</a>	<a href="#">E3</a>	<a href="#">HV</a>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				<a href="#">VMID</a>				

#### NS, bit [31]

Non-secure state sample. Indicates the Security state associated with the most recent [EDPCSR](#) sample.

If EL3 is not implemented, this bit indicates the Effective value of SCR.NS.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### E2, bit [30]

Exception level 2 status sample. Indicates whether the most recent [EDPCSR](#) sample was associated with EL2.

If EL2 is not implemented, this bit is RES0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### E3, bit [29]

Exception level 3 status sample. Indicates whether the most recent [EDPCSR](#) sample was associated with EL3 using AArch64.

If EDVIDSR.NS == 1 or the PE was in AArch32 state when EDPCSRlo ([EDPCSR](#)[31:0]) was read, this bit is 0.

If EL3 is not implemented, this bit is RES0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### HV, bit [28]

EDPCSRhi ([EDPCSR](#)[63:32]) valid. Indicates whether bits [63:32] of the most recent [EDPCSR](#) sample might be nonzero:

HV	Meaning
0	Bits[63:32] of the most recent EDPCSR sample are zero.
1	Bits[63:32] of the most recent EDPCSR sample might be nonzero.

An EDVIDSR.HV value of 1 does not mean that the value of EDPCSRhi is nonzero. An EDVIDSR.HV value of 0 is a hint that EDPCSRhi ([EDPCSR](#)[63:32]) does not need to be read.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### Bits [27:8]

Reserved, RES0.

### VMID, bits [7:0]

VMID sample. The VMID associated with the most recent EDPCSRlo ([EDPCSR](#)[31:0]) sample.

If the value of EDVIDSR.NS is 0 or the value of EDVIDSR.E2 is 1 this field is RES0.

If EL2 is not implemented, then this field is RES0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### When ARMv8.1-VHE is implemented and EDSCR.SC2 == 0:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NS	E2	E3	HV	0	0	0	0	0	0	0	0	0	0	0	0	VMID															

This format applies in all ARMv8.0 implementations.

### NS, bit [31]

Non-secure state sample. Indicates the Security state associated with the most recent [EDPCSR](#) sample.

If EL3 is not implemented, this bit indicates the Effective value of SCR.NS.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### E2, bit [30]

Exception level 2 status sample. Indicates whether the most recent [EDPCSR](#) sample was associated with EL2.

If EL2 is not implemented, this bit is RES0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**E3, bit [29]**

Exception level 3 status sample. Indicates whether the most recent [EDPCSR](#) sample was associated with EL3 using AArch64.

If EDVIDSR.NS == 1 or the PE was in AArch32 state when EDPCSRlo ([EDPCSR](#)[31:0]) was read, this bit is 0.

If EL3 is not implemented, this bit is RES0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**HV, bit [28]**

EDPCSRhi ([EDPCSR](#)[63:32]) valid. Indicates whether bits [63:32] of the most recent [EDPCSR](#) sample might be nonzero:

HV	Meaning
0	Bits[63:32] of the most recent EDPCSR sample are zero.
1	Bits[63:32] of the most recent EDPCSR sample might be nonzero.

An EDVIDSR.HV value of 1 does not mean that the value of EDPCSRhi is nonzero. An EDVIDSR.HV value of 0 is a hint that EDPCSRhi ([EDPCSR](#)[63:32]) does not need to be read.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Bits [27:16]**

Reserved, RES0.

**VMID, bits [15:0]**

VMID sample. The VMID associated with the most recent EDPCSRlo ([EDPCSR](#)[31:0]) sample.

- If the value of EDVIDSR.NS is 0 or the value of EDVIDSR.E2 is 1 this field is RES0.
- If EL2 is not implemented, this field is RES0.
- If EL2 is implemented and is using AArch64, the VMID is held in [VTTBR\\_EL2](#).VMID.
- If EL2 is implemented and is using AArch32, the VMID is held in [VTTBR](#).VMID.
- If 16-bit VMIDs are not supported, EDVIDSR.VMID[15:8] is RES0.
- If 16-bit VMIDs are supported, but VTTBRx.VMID[15:8] are not used, EDVIDSR.VMID[15:8] is set to 0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**When ARMv8.1-VHE is implemented and EDSCR.SC2 == 1:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<a href="#">CONTEXTIDR_EL2</a>																															

**CONTEXTIDR\_EL2, bits [31:0]**

Context ID.

- If EL2 is using AArch64, the value of CONTEXTIDR\_EL2 as associated with the most recent [EDPCSR](#) sample.
- If [EDPCSR](#).NS == 0, then this field is set to an UNKNOWN value.
- Otherwise this field is set to an UNKNOWN value.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**When ARMv8.2-PCSample is implemented:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
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

**Bits [31:0]**

Reserved, RES0.

## Accessing the EDVIDSR

EDVIDSR can be accessed through the external debug interface:

Component	Offset
Debug	0x0A8

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# EDWAR, External Debug Watchpoint Address Register

The EDWAR characteristics are:

## Purpose

Returns the virtual data address being accessed when a Watchpoint Debug Event was triggered.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	Default
Error	Error	Error	RO

## Configuration

EDWAR is in the Core power domain. RW 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.

## Attributes

EDWAR is a 64-bit register.

## Field descriptions

The EDWAR 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
Watchpoint address																															
Watchpoint address																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:0]

Watchpoint address. The data virtual address being accessed when a Watchpoint Debug Event was triggered and caused entry to Debug state. This address must be within a naturally-aligned block of memory of power-of-two size no larger than the [DC ZVA](#) block size.

The value of this register is UNKNOWN if the PE is in Non-debug state, or if Debug state was entered other than for a Watchpoint debug event.

The value of EDWAR[63:32] is UNKNOWN if Debug state was entered for a Watchpoint debug event taken from AArch32 state.

The EDWAR is subject to the same alignment rules as the reporting of a watchpointed address in the FAR. See 'Determining the memory location that caused a Watchpoint exception' 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.

## Accessing the EDWAR

EDWAR[31:0] can be accessed through the external debug interface:

Component	Offset
Debug	0x030



EDWAR[63:32] can be accessed through the external debug interface:

Component	Offset
Debug	0x034

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_ABPR, CPU Interface Aliased Binary Point Register

The GICC\_ABPR characteristics are:

## Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This register is used only when System register access is not enabled. When System register access is enabled, the System registers [ICC\\_BPR0\\_EL1](#) and [ICC\\_BPR1\\_EL1](#) provide equivalent functionality.

## Configuration

Some or all RW fields of this register have defined reset values.

In systems that support two Security states:

- This register is an alias of the Non-secure copy of [GICC\\_BPR](#).
- Non-secure accesses to this register return a shifted value of the binary point.
- If [ICC\\_CTLR\\_EL3](#).CBPR\_EL1NS == 1, Secure accesses to this register access [ICC\\_BPR0\\_EL1](#).

## Attributes

The reset value of this register is defined as (minimum [GICC\\_BPR](#).Binary\_Point + 1), resulting in a permitted range of 0×1-0×4.

## Field descriptions

The GICC\_ABPR 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	Binary_Point

### Bits [31:3]

Reserved, RES0.

### Binary\_Point, bits [2:0]

Controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. The following list describes how this field determines the interrupt priority bits assigned to the group priority field:

- Priority grouping for Group 1 interrupts when CBPR==0, for the processing of Group 1 interrupts in a GIC implementation that supports interrupt grouping, when [GICC\\_CTLR](#).CBPR == 0.
- Priority grouping for Group 0 interrupts, or Group 1 interrupts when CBPR==1, for all other cases.

When this register has an architecturally-defined reset value, this field resets to an architecturally UNKNOWN value.

## Accessing the GICC\_ABPR

GICC\_ABPR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x001C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_AEOIR, CPU Interface Aliased End Of Interrupt Register

The GICC\_AEOIR characteristics are:

## Purpose

A write to this register performs priority drop for the specified Group 1 interrupt and, if the appropriate [GICC\\_CTLR.EOImodeS](#) or [GICC\\_CTLR.EOImodeNS](#) field == 0, also deactivates the interrupt.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

A write to this register must correspond to the most recently acknowledged Group 1 interrupt. If a value other than the last value read from [GICC\\_AIAR](#) is written to this register, the effect is UNPREDICTABLE.

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_EOIR1](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_EOIR1\\_ELI](#) provides equivalent functionality.

When affinity routing is enabled for a Security state, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

When [GICD\\_CTLR.DS](#)==0, this register is an alias of the Non-secure view of [GICC\\_EOIR](#). A Secure access to this register is identical to a Non-secure access to [GICC\\_EOIR](#).

## Attributes

GICC\_AEOIR is a 32-bit register.

## Field descriptions

The GICC\_AEOIR 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		INTID																						

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the signaled interrupt.

#### Note

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

## Accessing the GICC\_AEOIR

GICC\_AEOIR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x0024

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_AHPPIR, CPU Interface Aliased Highest Priority Pending Interrupt Register

The GICC\_AHPPIR characteristics are:

## Purpose

If the highest priority pending interrupt is in Group 1, this register provides the INTID of the highest priority pending interrupt on the CPU interface.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_HPPIR1](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_HPPIR1\\_EL1](#) provides equivalent functionality.

If the highest priority pending interrupt is in Group 0, a read of this register returns the special INTID 1023.

Interrupt identifiers corresponding to an interrupt group that is not enabled are ignored.

If the highest priority pending interrupt is a direct interrupt that is both individually enabled in the Distributor and part of an interrupt group that is enabled in the Distributor, and the interrupt group is disabled in the CPU interface for this PE, this register returns the special INTID 1023.

See Preemption for more information about pending interrupts that are not considered when determining the highest priority pending interrupt.

When affinity routing is enabled for a Security state, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

If [GICD\\_CTLR.DS](#)==0, this register is an alias of the Non-secure view of [GICC\\_HPPIR](#). A Secure access to this register is identical to a Non-secure access to [GICC\\_HPPIR](#).

## Attributes

GICC\_AHPPIR is a 32-bit register.

## Field descriptions

The GICC\_AHPPIR 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													INTID											

INTID

### Bits [31:24]

Reserved, RES0.

**INTID, bits [23:0]**

The INTID of the signaled interrupt.

---

**Note**

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

---

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

## Accessing the GICC\_AHPPIR

GICC\_AHPPIR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x0028

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_AIAR, CPU Interface Aliased Interrupt Acknowledge Register

The GICC\_AIAR characteristics are:

## Purpose

Provides the INTID of the signaled Group 1 interrupt. A read of this register by the PE acts as an acknowledge for the interrupt.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

When affinity routing is enabled for a Security state, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

When [GICD\\_CTLR.DS==0](#), this register is an alias of the Non-secure view of [GICC\\_IAR](#). A Secure access to this register is identical to a Non-secure access to [GICC\\_IAR](#).

## Attributes

GICC\_AIAR is a 32-bit register.

## Field descriptions

The GICC\_AIAR 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																								

INTID

### Bits [31:24]

Reserved, RES0.

### INTID, bits [23:0]

The INTID of the signaled interrupt.

#### Note

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.



## Accessing the GICC\_AIAR

GICC\_AIAR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x0020 - 0x003C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_APR<n>, CPU Interface Active Priorities Registers, n = 0 - 3

The GICC\_APR<n> characteristics are:

## Purpose

Provides information about interrupt active priorities.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

These registers are used only when System register access is not enabled. When System register access is enabled the following registers provide equivalent functionality:

- In AArch64:
  - For Group 0, [ICC\\_AP0R<n>\\_EL1](#).
  - For Group 1, [ICC\\_AP1R<n>\\_EL1](#).
- In AArch32:
  - For Group 0, [ICC\\_AP0R<n>](#).
  - For Group 1, [ICC\\_AP1R<n>](#).

## Configuration

Some or all RW fields of this register have defined reset values.

The contents of these registers are IMPLEMENTATION DEFINED with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

When [GICD\\_CTLR.DS](#) == 0, these registers are Banked, and Non-secure accesses do not affect Secure operation. The Secure copies of these registers hold active priorities for Group 0 interrupts, and the Non-secure copies provide a Non-secure view of the active priorities for Group 1 interrupts.

GICC\_APR1 is only implemented in implementations that support 6 or more bits of priority. GICC\_APR2 and GICC\_APR3 are only implemented in implementations that support 7 bits of priority.

When [GICD\\_CTLR.DS](#) == 1, these registers hold the active priorities for Group 0 interrupts, and the active priorities for Group 1 interrupts are held by the [GICC\\_NSAPR<n>](#) registers.

## Attributes

GICC\_APR<n> is a 32-bit register.

## Field descriptions

The GICC\_APR<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
IMPLEMENTATION DEFINED																															

**IMPLEMENTATION DEFINED, bits [31:0]**

IMPLEMENTATION DEFINED.

When this register has an architecturally-defined reset value, this field resets to 0.

**Accessing the GICC\_APR<n>**

GICC\_APR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	$0 \times 00D0 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_BPR, CPU Interface Binary Point Register

The GICC\_BPR characteristics are:

## Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This register is used only when System register access is not enabled. When System register access is enabled this register is RAZ/WI, and the System registers [ICC\\_BPR0\\_EL1](#) and [ICC\\_BPR1\\_EL1](#) provide equivalent functionality.

## Configuration

Some or all RW fields of this register have defined reset values.

In systems that support two Security states:

- This register is Banked.
- The Secure instance of this register determines Group 0 interrupt preemption.
- The Non-secure instance of this register determines Group 1 interrupt preemption.

In systems that support only one Security state, when [GICC\\_CTLR](#).CBPR == 0, this register determines only Group 0 interrupt preemption.

When [GICC\\_CTLR](#).CBPR == 1, this register determines interrupt preemption for both Group 0 and Group 1 interrupts.

## Attributes

GICC\_BPR is a 32-bit register.

## Field descriptions

The GICC\_BPR 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	Binary_Point

### Bits [31:3]

Reserved, RES0.

### Binary\_Point, bits [2:0]

Controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. The following list describes how this field determines the interrupt priority bits assigned to the group priority field:

- Priority grouping for Group 1 interrupts when CBPR==0, for the processing of Group 1 interrupts in a GIC implementation that supports interrupt grouping, when [GICC\\_CTLR](#).CBPR == 0.
- Priority grouping for Group 0 interrupts, or Group 1 interrupts when CBPR==1, for all other cases.

When this register has an architecturally-defined reset value, this field resets to an architecturally UNKNOWN value.

---

**Note**

Aliasing the Non-secure GICC\_BPR as [GICC\\_ABPR](#) in a multiprocessor system permits a PE that can make only Secure accesses to configure the preemption setting for Group 1 interrupts by accessing [GICC\\_ABPR](#).

---

## Accessing the GICC\_BPR

GICC\_BPR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x0008

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_CTLR, CPU Interface Control Register

The GICC\_CTLR characteristics are:

## Purpose

Controls the CPU interface, including enabling of interrupt groups, interrupt signal bypass, binary point registers used, and separation of priority drop and interrupt deactivation.

### Note

If the GIC implementation supports two Security states, independent EOI controls are provided for accesses from each Security state. Secure accesses handle both Group 0 and Group 1 interrupts, and Non-secure accesses handle Group 1 interrupts only.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_CTLR](#) and [ICC\\_MCTLR](#) provide equivalent functionality.
- For AArch64 implementations, [ICC\\_CTLR\\_EL1](#) and [ICC\\_CTLR\\_EL3](#) provide equivalent functionality.

## Configuration

Some or all RW fields of this register have defined reset values.

In a GIC implementation that supports two Security states:

- This register is Banked.
- The register bit assignments are different in the Secure and Non-secure copies.

## Attributes

GICC\_CTLR is a 32-bit register.

## Field descriptions

The GICC\_CTLR bit assignments are:

### When GICD\_CTLR.DS==0, Non-secure access:

31	30	29	28	27	26	25	24	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	EOImodeNS	0	0	IRQBypDisGrp1	FIQBypDisGrp1	0	0	0	0	EnableGrp1

### Bits [31:10]

Reserved, RES0.

**EOImodeNS, bit [9]**

Controls the behavior of Non-secure accesses to [GICC\\_EOIR](#), [GICC\\_AEOIR](#), and [GICC\\_DIR](#).

EOImodeNS	Meaning
0	<a href="#">GICC_EOIR</a> and <a href="#">GICC_AEOIR</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">GICC_DIR</a> are UNPREDICTABLE.
1	<a href="#">GICC_EOIR</a> and <a href="#">GICC_AEOIR</a> provide priority drop functionality only. <a href="#">GICC_DIR</a> provides interrupt deactivation functionality.

**Note**

An implementation is permitted to make this bit RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**Bits [8:7]**

Reserved, RES0.

**IRQBypDisGrp1, bit [6]**

When the signaling of IRQs by the CPU interface is disabled, this field partly controls whether the bypass IRQ signal is signaled to the PE for Group 1:

IRQBypDisGrp1	Meaning
0	The bypass IRQ signal is signaled to the PE.
1	The bypass IRQ signal is not signaled to the PE.

If System register access is enabled for EL3 and [ICC\\_SRE\\_EL3](#).DIB == 1, this field is RAO/WI.

If System register access is enabled for EL1, this field is ignored.

If an implementation does not support legacy interrupts, this bit is permitted to be RAO/WI.

See Interrupt signal bypass and bypass disable for more information.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**FIQBypDisGrp1, bit [5]**

When the signaling of FIQs by the CPU interface is disabled, this field partly controls whether the bypass FIQ signal is signaled to the PE for Group 1:

FIQBypDisGrp1	Meaning
0	The bypass FIQ signal is signaled to the PE.
1	The bypass FIQ signal is not signaled to the PE.

If System register access is enabled for EL3 and [ICC\\_SRE\\_EL3](#).DFB == 1, this field is RAO/WI.

If System register access is enabled for EL1, this field is ignored.

If an implementation does not support legacy interrupts, this bit is permitted to be RAO/WI.

See Interrupt signal bypass and bypass disable for more information.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**Bits [4:1]**

Reserved, RES0.

**EnableGrp1, bit [0]**

This Non-secure field enables the signaling of Group 1 interrupts by the CPU interface to a target PE:

EnableGrp1	Meaning
0	Group 1 interrupt signaling is disabled.
1	Group 1 interrupt signaling is enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

**When GICD\_CTLR.DS==0, Secure access:**

313029282726252423222120191817161514131211	10	9	8	7	6	5
000000000000000000000000	EOImodeNS	EOImodeS	IRQBypDisGrp1	FIQBypDisGrp1	IRQBypDisGrp0	FIQBypDisGrp0

**Bits [31:11]**

Reserved, RES0.

**EOImodeNS, bit [10]**

Controls the behavior of Non-secure accesses to [GICC\\_EOIR](#), [GICC\\_AEOIR](#), and [GICC\\_DIR](#).

EOImodeNS	Meaning
0	<a href="#">GICC_EOIR</a> and <a href="#">GICC_AEOIR</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">GICC_DIR</a> are UNPREDICTABLE.
1	<a href="#">GICC_EOIR</a> and <a href="#">GICC_AEOIR</a> provide priority drop functionality only. <a href="#">GICC_DIR</a> provides interrupt deactivation functionality.

**Note**

An implementation is permitted to make this bit RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**EOImodeS, bit [9]**

Controls the behavior of Secure accesses to [GICC\\_EOIR](#), [GICC\\_AEOIR](#), and [GICC\\_DIR](#).

EOImodeS	Meaning
0	<a href="#">GICC_EOIR</a> and <a href="#">GICC_AEOIR</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">GICC_DIR</a> are UNPREDICTABLE.
1	<a href="#">GICC_EOIR</a> and <a href="#">GICC_AEOIR</a> provide priority drop functionality only. <a href="#">GICC_DIR</a> provides interrupt deactivation functionality.

**Note**

An implementation is permitted to make this bit RAO/WI.

This field shares state with [GICC\\_CTLR.EOImode](#).

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**IRQBypDisGrp1, bit [8]**

When the signaling of IRQs by the CPU interface is disabled, this field partly controls whether the bypass IRQ signal is signaled to the PE for Group 1:

IRQBypDisGrp1	Meaning
0	The bypass IRQ signal is signaled to the PE.
1	The bypass IRQ signal is not signaled to the PE.



If System register access is enabled for EL3 and [ICC\\_SRE\\_EL3](#).DIB == 1, this field is RAO/WI.

If System register access is enabled for EL1, this field is ignored.

If an implementation does not support legacy interrupts, this bit is permitted to be RAO/WI.

See Interrupt signal bypass and bypass disable for more information.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### FIQBypDisGrp1, bit [7]

When the signaling of FIQs by the CPU interface is disabled, this field partly controls whether the bypass FIQ signal is signaled to the PE for Group 1:

FIQBypDisGrp1	Meaning
0	The bypass FIQ signal is signaled to the PE.
1	The bypass FIQ signal is not signaled to the PE.

If System register access is enabled for EL3 and [ICC\\_SRE\\_EL3](#).DFB == 1, this field is RAO/WI.

If System register access is enabled for EL1, this field is ignored.

If an implementation does not support legacy interrupts, this bit is permitted to be RAO/WI.

See Interrupt signal bypass and bypass disable for more information.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### IRQBypDisGrp0, bit [6]

When the signaling of IRQs by the CPU interface is disabled, this field partly controls whether the bypass IRQ signal is signaled to the PE for Group 0:

IRQBypDisGrp0	Meaning
0	The bypass IRQ signal is signaled to the PE.
1	The bypass IRQ signal is not signaled to the PE.

If System register access is enabled for EL3 and [ICC\\_SRE\\_EL3](#).DIB == 1, this field is RAO/WI.

If System register access is enabled for EL1, this field is ignored.

If an implementation does not support legacy interrupts, this bit is permitted to be RAO/WI.

See Interrupt signal bypass and bypass disable for more information.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### FIQBypDisGrp0, bit [5]

When the signaling of FIQs by the CPU interface is disabled, this field partly controls whether the bypass FIQ signal is signaled to the PE for Group 0:

FIQBypDisGrp0	Meaning
0	The bypass FIQ signal is signaled to the PE.
1	The bypass FIQ signal is not signaled to the PE.

If System register access is enabled for EL3 and [ICC\\_SRE\\_EL3](#).DIB == 1, this field is RAO/WI.

If System register access is enabled for EL1, this field is ignored.

If an implementation does not support legacy interrupts, this bit is permitted to be RAO/WI.

See Interrupt signal bypass and bypass disable for more information.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**CBPR, bit [4]**

Controls whether [GICC\\_BPR](#) provides common control of preemption to Group 0 and Group 1 interrupts:

CBPR	Meaning
0	<a href="#">GICC_BPR</a> determines preemption for Group 0 interrupts only. <a href="#">GICC_ABPR</a> determines preemption for Group 1 interrupts.
1	<a href="#">GICC_BPR</a> determines preemption for both Group 0 and Group 1 interrupts.

This field is an alias of [ICC\\_CTLR\\_EL3.CBPR\\_EL1NS](#).

In a GIC that supports two Security states, when CBPR == 1:

- A Non-secure read of [GICC\\_BPR](#) returns the value of Secure [GICC\\_BPR](#).BinaryPoint, incremented by 1, and saturated to 0b1111.
- Non-secure writes of [GICC\\_BPR](#) are ignored.

When this register has an architecturally-defined reset value, this field resets to 0.

**FIQEn, bit [3]**

Controls whether the CPU interface signals Group 0 interrupts to a target PE using the FIQ or IRQ signal:

FIQEn	Meaning
0	Group 0 interrupts are signaled using the IRQ signal.
1	Group 0 interrupts are signaled using the FIQ signal.

Group 1 interrupts are signaled using the IRQ signal only.

If an implementation supports two Security states, this bit is permitted to be RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**Bit [2]**

Reserved, RES0.

**EnableGrp1, bit [1]**

This Non-secure field enables the signaling of Group 1 interrupts by the CPU interface to a target PE:

EnableGrp1	Meaning
0	Group 1 interrupt signaling is disabled.
1	Group 1 interrupt signaling is enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

**EnableGrp0, bit [0]**

Enables the signaling of Group 0 interrupts by the CPU interface to a target PE:

EnableGrp0	Meaning
0	Group 0 interrupt signaling is disabled.
1	Group 0 interrupt signaling is enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

**When GICD\_CTLR.DS==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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EOImode	IRQBypDisGrp1	FIQBypDisGrp1	IRQBypDisGrp0	FIQBypDisGrp0	CBPRFIQ

**Bits [31:10]**

Reserved, RES0.

**EOImode, bit [9]**

Controls the behavior of accesses to [GICC\\_EOIR](#), [GICC\\_AEOIR](#), and [GICC\\_DIR](#).

EOImode	Meaning
0	<a href="#">GICC_EOIR</a> and <a href="#">GICC_AEOIR</a> provide both priority drop and interrupt deactivation functionality. Accesses to <a href="#">GICC_DIR</a> are UNPREDICTABLE.
1	<a href="#">GICC_EOIR</a> and <a href="#">GICC_AEOIR</a> provide priority drop functionality only. <a href="#">GICC_DIR</a> provides interrupt deactivation functionality.

**Note**

An implementation is permitted to make this bit RAO/WI.

This field shares state with [GICC\\_CTLR.EOImodeS](#).

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**IRQBypDisGrp1, bit [8]**

When the signaling of IRQs by the CPU interface is disabled, this field partly controls whether the bypass IRQ signal is signaled to the PE for Group 1:

IRQBypDisGrp1	Meaning
0	The bypass IRQ signal is signaled to the PE.
1	The bypass IRQ signal is not signaled to the PE.

If System register access is enabled for EL3 and [ICC\\_SRE\\_EL3.DIB](#) == 1, this field is RAO/WI.

If System register access is enabled for EL1, this field is ignored.

If an implementation does not support legacy interrupts, this bit is permitted to be RAO/WI.

See Interrupt signal bypass and bypass disable for more information.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**FIQBypDisGrp1, bit [7]**

When the signaling of FIQs by the CPU interface is disabled, this field partly controls whether the bypass FIQ signal is signaled to the PE for Group 1:

FIQBypDisGrp1	Meaning
0	The bypass FIQ signal is signaled to the PE.
1	The bypass FIQ signal is not signaled to the PE.

If System register access is enabled for EL3 and [ICC\\_SRE\\_EL3.DFB](#) == 1, this field is RAO/WI.

If System register access is enabled for EL1, this field is ignored.

If an implementation does not support legacy interrupts, this bit is permitted to be RAO/WI.

See Interrupt signal bypass and bypass disable for more information.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**IRQBypDisGrp0, bit [6]**

When the signaling of IRQs by the CPU interface is disabled, this field partly controls whether the bypass IRQ signal is signaled to the PE for Group 0:

IRQBypDisGrp0	Meaning
0	The bypass IRQ signal is signaled to the PE.
1	The bypass IRQ signal is not signaled to the PE.

If System register access is enabled for EL3 and [ICC\\_SRE\\_EL3.DIB](#) == 1, this field is RAO/WI.

If System register access is enabled for EL1, this field is ignored.

If an implementation does not support legacy interrupts, this bit is permitted to be RAO/WI.

See Interrupt signal bypass and bypass disable for more information.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### FIQBypDisGrp0, bit [5]

When the signaling of FIQs by the CPU interface is disabled, this field partly controls whether the bypass FIQ signal is signaled to the PE for Group 0:

FIQBypDisGrp0	Meaning
0	The bypass FIQ signal is signaled to the PE.
1	The bypass FIQ signal is not signaled to the PE.

If System register access is enabled for EL3 and [ICC\\_SRE\\_EL3.DIB](#) == 1, this field is RAO/WI.

If System register access is enabled for EL1, this field is ignored.

If an implementation does not support legacy interrupts, this bit is permitted to be RAO/WI.

See Interrupt signal bypass and bypass disable for more information.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### CBPR, bit [4]

Controls whether [GICC\\_BPR](#) provides common control of preemption to Group 0 and Group 1 interrupts:

CBPR	Meaning
0	<a href="#">GICC_BPR</a> determines preemption for Group 0 interrupts only. <a href="#">GICC_ABPR</a> determines preemption for Group 1 interrupts.
1	<a href="#">GICC_BPR</a> determines preemption for both Group 0 and Group 1 interrupts.

This field is an alias of [ICC\\_CTLR\\_EL3.CBPR\\_EL1NS](#).

In a GIC that supports two Security states, when CBPR == 1:

- A Non-secure read of [GICC\\_BPR](#) returns the value of Secure [GICC\\_BPR](#).BinaryPoint, incremented by 1, and saturated to 0b111.
- Non-secure writes of [GICC\\_BPR](#) are ignored.

When this register has an architecturally-defined reset value, this field resets to 0.

#### FIQEn, bit [3]

Controls whether the CPU interface signals Group 0 interrupts to a target PE using the FIQ or IRQ signal:

FIQEn	Meaning
0	Group 0 interrupts are signaled using the IRQ signal.
1	Group 0 interrupts are signaled using the FIQ signal.

Group 1 interrupts are signaled using the IRQ signal only.

If an implementation supports two Security states, this bit is permitted to be RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**Bit [2]**

Reserved, RES0.

**EnableGrp1, bit [1]**

This Non-secure field enables the signaling of Group 1 interrupts by the CPU interface to a target PE:

EnableGrp1	Meaning
0	Group 1 interrupt signaling is disabled.
1	Group 1 interrupt signaling is enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

**EnableGrp0, bit [0]**

Enables the signaling of Group 0 interrupts by the CPU interface to a target PE:

EnableGrp0	Meaning
0	Group 0 interrupt signaling is disabled.
1	Group 0 interrupt signaling is enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

**Accessing the GICC\_CTLR**

GICC\_CTLR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x0000

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_DIR, CPU Interface Deactivate Interrupt Register

The GICC\_DIR characteristics are:

## Purpose

When interrupt priority drop is separated from interrupt deactivation, a write to this register deactivates the specified interrupt.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_DIR](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_DIR\\_EL1](#) provides equivalent functionality.

Writes to this register have an effect only in the following cases:

- When [GICD\\_CTLR](#).DS == 1, if [GICC\\_CTLR](#).EOImode == 1.
- In GIC implementations that support two Security states:
  - If the access is Secure and [GICC\\_CTLR](#).EOImodeS == 1.
  - If the access is Non-secure and [GICC\\_CTLR](#).EOImodeNS == 1.

The following writes must be ignored:

- Writes to this register when the corresponding EOImode field in [GICC\\_CTLR](#) == 0. In systems that support system error generation, an implementation might generate a system error.
- Writes to this register when the corresponding EOImode field in [GICC\\_CTLR](#) == 0 and the corresponding interrupt is not active. In systems that support system error generation, an implementation might generate a system error. In implementations using the GIC Stream Protocol Interface these writes correspond to a Deactivate packet for an interrupt that is not active.

If the corresponding EOImode field in [GICC\\_CTLR](#) is 1 and this register is written to without a corresponding write to [GICC\\_EOIR](#) or [GICC\\_AEOIR](#), the interrupt is deactivated but the bit corresponding to it in the active priorities registers remains set.

When affinity routing is enabled for a Security state, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

There are no configuration notes.

## Attributes

GICC\_DIR is a 32-bit register.

## Field descriptions

The GICC\_DIR 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																								

INTID

**Bits [31:24]**

Reserved, RES0.

**INTID, bits [23:0]**

The INTID of the signaled interrupt.

**Note**

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

**Accessing the GICC\_DIR**

GICC\_DIR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x1000

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_EOIR, CPU Interface End Of Interrupt Register

The GICC\_EOIR characteristics are:

## Purpose

A write to this register performs priority drop for the specified interrupt and, if the appropriate [GICC\\_CTLR.EOImodeS](#) or [GICC\\_CTLR.EOImodeNS](#) field == 0, also deactivates the interrupt.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

The following writes must be ignored:

- Writes of INTIDs 1020-1023.
- Secure writes corresponding to Group 1 interrupts. In systems that support system error generation, an implementation might generate a system error. In this case, GIC behavior is predictable, and the highest Secure active priority (in the Secure copy of [GICC\\_APR<n>](#)) will be reset if the highest active priority is Secure. System behavior is UNPREDICTABLE.
- Non-secure writes corresponding to Group 0 interrupts when [GICC\\_CTLR.EOImodeS](#) == 1. In systems that support system error generation, an implementation might generate a system error. In this case, GIC behavior is predictable, and the highest Non-secure active priority (in the Non-secure copy of [GICC\\_APR<n>](#)) will be reset if the highest active priority is Non-secure. System behavior is UNPREDICTABLE.

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_EOIR0](#) and [ICC\\_EOIR1](#) provide equivalent functionality.
- For AArch64 implementations, [ICC\\_EOIR0\\_EL1](#) and [ICC\\_EOIR1\\_EL1](#) provide equivalent functionality.

When affinity routing is enabled for a Security state, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

If [GICD\\_CTLR.DS](#)==0:

- This register is Common.
- [GICC\\_AEOIR](#) is an alias of the Non-secure view of this register.

For Secure writes when [GICD\\_CTLR.DS](#)==0, or for Secure and Non-secure writes when [GICD\\_CTLR.DS](#)==1, the register provides functionality for Group 0 interrupts.

For Non-secure writes when [GICD\\_CTLR.DS](#)==1, the register provides functionality for Group 1 interrupts.

## Attributes

GICC\_EOIR is a 32-bit register.

## Field descriptions

The GICC\_EOIR 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	INTID																										



Bits [31:24]

Reserved, RES0.

INTID, bits [23:0]

The INTID of the signaled interrupt.

Note

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

For every read of a valid INTID from [GICC\\_IAR](#), the connected PE must perform a matching write to GICC\_EOIR. The value written to GICC\_EOIR must be the INTID from [GICC\\_IAR](#). Reads of INTIDs 1020-1023 do not require matching writes.

Note

ARM recommends that software preserves the entire register value read from [GICC\\_IAR](#), and writes that value back to GICC\_EOIR on completion of interrupt processing.

For nested interrupts, the order of writes to this register must be the reverse of the order of interrupt acknowledgement. Behavior is UNPREDICTABLE if:

- This ordering constraint is not maintained.
- The value written to this register does not match an active interrupt, or the ID of a spurious interrupt.
- The value written to this register does not match the last valid interrupt value read from [GICC\\_IAR](#).

See Interrupt lifecycle for general information about the effect of writes to end of interrupt registers, and about the possible separation of the priority drop and interrupt deactivate operations.

If [GICD\\_CTLR](#).DS==0:

- [GICC\\_CTLR](#).EOImodeS controls the behavior of Secure accesses to GICC\_EOIR and [GICC\\_AEOIR](#).
- [GICC\\_CTLR](#).EOImodeNS controls the behavior of Non-secure accesses to GICC\_EOIR and [GICC\\_AEOIR](#).

Accessing the GICC\_EOIR

GICC\_EOIR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x0010

# GICC\_HPPIR, CPU Interface Highest Priority Pending Interrupt Register

The GICC\_HPPIR characteristics are:

## Purpose

Provides the INTID of the highest priority pending interrupt on the CPU interface.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_HPPIR0](#) and [ICC\\_HPPIR1](#) provide equivalent functionality.
- For AArch64 implementations, [ICC\\_HPPIR0\\_EL1](#) and [ICC\\_HPPIR1\\_EL1](#) provide equivalent functionality.

If the highest priority pending interrupt is in Group 0, a Non-secure read of this register returns the special INTID 1023.

For Secure reads when [GICD\\_CTLR](#).DS==0, or for Secure and Non-secure reads when [GICD\\_CTLR](#).DS==1, returns the special INTID 1022 if the highest priority pending interrupt is in Group 1.

If no interrupts are in the pending state, a read of this register returns the special INTID 1023.

Interrupt identifiers corresponding to an interrupt group that is not enabled are ignored.

If the highest priority pending interrupt is a direct interrupt that is both individually enabled in the Distributor and part of an interrupt group that is enabled in the Distributor, and the interrupt group is disabled in the CPU interface for this PE, this register returns the special INTID 1023.

See Preemption for more information about pending interrupts that are not considered when determining the highest priority pending interrupt.

When affinity routing is enabled for a Security state, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

If [GICD\\_CTLR](#).DS==0:

- This register is Common.
- [GICC\\_AHPPIR](#) is an alias of the Non-secure view of this register.

## Attributes

GICC\_HPPIR is a 32-bit register.

## Field descriptions

The GICC\_HPPIR 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	INTID																											

**Bits [31:24]**

Reserved, RES0.

**INTID, bits [23:0]**

The INTID of the signaled interrupt.

---

**Note**

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

---

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

## Accessing the GICC\_HPPIR

GICC\_HPPIR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x0018

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_IAR, CPU Interface Interrupt Acknowledge Register

The GICC\_IAR characteristics are:

## Purpose

Provides the INTID of the signaled interrupt. A read of this register by the PE acts as an acknowledge for the interrupt.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

When [GICD\\_CTLR.DS](#)=1, if the highest priority pending interrupt is in Group 1, the special INTID 1022 is returned.

In GIC implementations that support two Security states, if the highest priority pending interrupt is in Group 0, Non-secure reads return the special INTID 1023.

In GIC implementations that support two Security states, if the highest priority pending interrupt is in Group 1, Secure reads return the special INTID 1022.

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_IAR0](#) and [ICC\\_IAR1](#) provide equivalent functionality.
- For AArch64 implementations, [ICC\\_IAR0\\_EL1](#) and [ICC\\_IAR1\\_EL1](#) provide equivalent functionality.

When affinity routing is enabled for a Security state, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

This register is available in all configurations of the GIC. If [GICD\\_CTLR.DS](#)=0:

- This register is Common.
- [GICC\\_AIAR](#) is an alias of the Non-secure view of this register.

The format of the INTID is governed by whether affinity routing is enabled for a Security state.

## Attributes

GICC\_IAR is a 32-bit register.

## Field descriptions

The GICC\_IAR 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																								

INTID

### Bits [31:24]

Reserved, RES0.

**INTID, bits [23:0]**

The INTID of the signaled interrupt.

**Note**

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

A read of this register returns the INTID of the highest priority pending interrupt for the CPU interface. The read returns a spurious INTID of 1023 if any of the following apply:

- Forwarding of interrupts by the Distributor to the CPU interface is disabled.
- Signaling of interrupts by the CPU interface to the connected PE is disabled.
- There are no pending interrupts on the CPU interface with sufficient priority for the interface to signal it to the PE.

When the GIC returns a valid INTID to a read of this register it treats the read as an acknowledge of that interrupt. In addition, it changes the interrupt status from pending to active, or to active and pending if the pending state of the interrupt persists. Normally, the pending state of an interrupt persists only if the interrupt is level-sensitive and remains asserted.

For every read of a valid INTID from GICC\_IAR, the connected PE must perform a matching write to [GICC\\_EOIR](#).

**Note**

- ARM recommends that software preserves the entire register value read from this register, and writes that value back to [GICC\\_EOIR](#) on completion of interrupt processing.
- For SPIs, although multiple target PEs might attempt to read this register at any time, only one PE can obtain a valid INTID. See 'Interrupt acknowledgement', section 4.7.1 of the GICv3 Architecture Specification, for more information.

**Accessing the GICC\_IAR**

GICC\_IAR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x000C

# GICC\_IIDR, CPU Interface Identification Register

The GICC\_IIDR characteristics are:

## Purpose

Provides information about the implementer and revision of the CPU interface.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

## Configuration

There are no configuration notes.

## Attributes

GICC\_IIDR is a 32-bit register.

## Field descriptions

The GICC\_IIDR 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
ProductID												Architecture_version			Revision			Implementer													

### ProductID, bits [31:20]

An IMPLEMENTATION DEFINED product identifier.

### Architecture\_version, bits [19:16]

The version of the GIC architecture that is implemented.

Architecture_version	Meaning
0001	GICv1.
0010	GICv2.
0011	GICv3 memory-mapped interface supported. Support for the System register interface is discoverable from PE registers ID_PFR1 and ID_AA64PFR0_EL1.
0100	GICv4 memory-mapped interface supported. Support for the System register interface is discoverable from PE registers ID_PFR1 and ID_AA64PFR0_EL1.

Other values are reserved.

### Revision, bits [15:12]

An IMPLEMENTATION DEFINED revision number for the CPU interface.

**Implementer, bits [11:0]**

Contains the JEP106 code of the company that implemented the CPU interface.

- Bits [11:8] are the JEP106 continuation code of the implementer. For an ARM implementation, this field is 0x4.
- Bit [7] is always 0.
- Bits [6:0] are the JEP106 identity code of the implementer. For an ARM implementation, bits [7:0] are therefore 0x3B.

## Accessing the GICC\_IIDR

GICC\_IIDR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x00FC

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_NSAPR<n>, CPU Interface Non-secure Active Priorities Registers, n = 0 - 3

The GICC\_NSAPR<n> characteristics are:

## Purpose

Provides information about Group 1 interrupt active priorities.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

## Configuration

Some or all RW fields of this register have defined reset values.

The contents of these registers are IMPLEMENTATION DEFINED with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

When [GICD\\_CTLR.DS](#)==0, these registers are RAZ/WI to Non-secure accesses.

GICC\_NSAPR1 is only implemented in implementations that support 6 or more bits of priority. GICC\_NSAPR2 and GICC\_NSAPR3 are only implemented in implementations that support 7 bits of priority.

## Attributes

GICC\_NSAPR<n> is a 32-bit register.

## Field descriptions

The GICC\_NSAPR<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
IMPLEMENTATION DEFINED																															

### IMPLEMENTATION DEFINED, bits [31:0]

IMPLEMENTATION DEFINED.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the GICC\_NSAPR<n>

GICC\_NSAPR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x00E0 + 4n





# GICC\_PMR, CPU Interface Priority Mask Register

The GICC\_PMR characteristics are:

## Purpose

This register provides an interrupt priority filter. Only interrupts with a higher priority than the value in this register are signaled to the PE.

**Note**

Higher interrupt priority corresponds to a lower value of the Priority field.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

If the GIC implementation supports two Security states:

- Non-secure accesses to this register can only read or write values corresponding to the lower half of the priority range.
- If a Secure write has programmed the register with a value that corresponds to a value in the upper half of the priority range then:
  - Any Non-secure read of the register returns 0x00, regardless of the value held in the register.
  - Non-secure writes are ignored.

See 'Priority control of Secure and Non-secure interrupts' in the GICv3 Architecture Specification for more information.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

This register is available in all configurations of the GIC. If the GIC implementation supports two Security states this register is Common.

## Attributes

GICC\_PMR is a 32-bit register.

## Field descriptions

The GICC\_PMR 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	Priority									

**Bits [31:8]**

Reserved, RES0.

**Priority, bits [7:0]**

The priority mask level for the CPU interface. If the priority of the interrupt is higher than the value indicated by this field, the interface signals the interrupt to the PE.

If the GIC implementation supports fewer than 256 priority levels some bits might be RAZ/WI, as follows:

- For 128 supported levels, bit [0] = 0b0.
- For 64 supported levels, bits [1:0] = 0b00.
- For 32 supported levels, bits [2:0] = 0b000.
- For 16 supported levels, bits [3:0] = 0b0000.

See Interrupt prioritization for more information.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICC\_PMR

GICC\_PMR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x0004

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_RPR, CPU Interface Running Priority Register

The GICC\_RPR characteristics are:

## Purpose

This register indicates the running priority of the CPU interface.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

If there is no active interrupt on the CPU interface, the idle priority value is returned.

If the GIC implementation supports two Security states, a Non-secure read of the Priority field returns:

- 0x00 if the field value is less than 0x80.
- The Non-secure view of the Priority value if the field value is 0x80 or more.

See 'Priority control of Secure and Non-secure interrupts' in the GICv3 Architecture Specification for more information.

---

### Note

Software cannot determine the number of implemented priority bits from this register.

---

## Configuration

This register is available in all configurations of the GIC. If the GIC implementation supports two Security states this register is Common.

## Attributes

GICC\_RPR is a 32-bit register.

## Field descriptions

The GICC\_RPR 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	Priority							

### Bits [31:8]

Reserved, RES0.

### Priority, bits [7:0]

The current running priority on the CPU interface.

## Accessing the GICC\_RPR

GICC\_RPR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x0014

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICC\_STATUSR, CPU Interface Status Register

The GICC\_STATUSR characteristics are:

## Purpose

Provides software with a mechanism to detect:

- Accesses to reserved locations.
- Writes to read-only locations.
- Reads of write-only locations.

This register is part of the GIC physical CPU interface registers functional group.

## Usage constraints

In an implementation that supports two Security states, there are separate Secure and Non-secure instances of this register:

	Security disabled	Secure	Non-secure
GICC_STATUSR(S)	RW	RW	-
GICC_STATUSR(NS)	RW	-	RW

This is an optional register. If the register is not implemented, the location is RAZ/WI.

If this register is implemented, [GICV\\_STATUSR](#) must also be implemented.

## Configuration

If the GIC implementation supports two Security states this register is Banked to provide Secure and Non-secure copies.

This register is used only when System register access is not enabled. If System register access is enabled, this register is not updated. Equivalent functionality might be provided by appropriate traps and exceptions.

## Attributes

GICC\_STATUSR is a 32-bit register.

## Field descriptions

The GICC\_STATUSR 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	ASV	WROD	RWOD	WRD	RRD

### Bits [31:5]

Reserved, RES0.

### ASV, bit [4]

Attempted security violation.

ASV	Meaning
0	Normal operation.
1	A Non-secure access to a Secure register has been detected.

**Note**

This bit is not set to 1 for registers where any of the fields are Non-secure.

**WROD, bit [3]**

Write to an RO location.

WROD	Meaning
0	Normal operation.
1	A write to an RO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**RWOD, bit [2]**

Read of a WO location.

RWOD	Meaning
0	Normal operation.
1	A read of a WO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**WRD, bit [1]**

Write to a reserved location.

WRD	Meaning
0	Normal operation.
1	A write to a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**RRD, bit [0]**

Read of a reserved location.

RRD	Meaning
0	Normal operation.
1	A read of a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**Accessing the GICC\_STATUSR**

GICC\_STATUSR can be accessed through its memory-mapped interface:

Component	Offset
GIC CPU interface	0x002C

# GICD\_CLRSPI\_NSR, Clear Non-secure SPI Pending Register

The GICD\_CLRSPI\_NSR characteristics are:

## Purpose

Removes the pending state from a valid SPI if permitted by the Security state of the access and the [GICD\\_NSACR<n>](#) value for that SPI.

A write to this register changes the state of a pending SPI to inactive, and the state of an active and pending SPI to active.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

Writes to this register have no effect if:

- The value written specifies a Secure SPI, the value is written by a Non-secure access, and the value of the corresponding [GICD\\_NSACR<n>](#) register is less than 0b10.
- The value written specifies an invalid SPI.
- The SPI is not pending.

16-bit accesses to bits [15:0] of this register must be supported.

---

### Note

A Secure access to this register can clear the pending state of any valid SPI.

---

## Configuration

If [GICD\\_TYPER](#).MBIS == 0, this register is reserved.

When [GICD\\_CTLR](#).DS==1, this register provides functionality for all SPIs.

## Attributes

GICD\_CLRSPI\_NSR is a 32-bit register.

## Field descriptions

The GICD\_CLRSPI\_NSR 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	INTID															

[INTID](#)

### Bits [31:10]

Reserved, RES0.



**INTID, bits [9:0]**

The INTID of the SPI.

The function of this register depends on whether the targeted SPI is configured to be an edge-triggered or level-sensitive interrupt:

- For an edge-triggered interrupt, a write to [GICD\\_SETSPI\\_NSR](#) or [GICD\\_SETSPI\\_SR](#) adds the pending state to the targeted interrupt. It will stop being pending on activation, or if the pending state is removed by a write to GICD\_CLRSPI\_NSR, [GICD\\_CLRSPI\\_SR](#), or [GICD\\_ICPENDR<n>](#).
- For a level-sensitive interrupt, a write to [GICD\\_SETSPI\\_NSR](#) or [GICD\\_SETSPI\\_SR](#) adds the pending state to the targeted interrupt. It will remain pending until it is deasserted by a write to GICD\_CLRSPI\_NSR or [GICD\\_CLRSPI\\_SR](#). If the interrupt is activated between having the pending state added and being deactivated, then the interrupt will be active and pending.

## Accessing the GICD\_CLRSPI\_NSR

GICD\_CLRSPI\_NSR can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	0x0048

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_CLRSPI\_SR, Clear Secure SPI Pending Register

The GICD\_CLRSPI\_SR characteristics are:

## Purpose

Removes the pending state from a valid SPI.

A write to this register changes the state of a pending SPI to inactive, and the state of an active and pending SPI to active.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WI	WO	WI

Writes to this register have no effect if:

- The value is written by a Non-secure access.
- The value written specifies an invalid SPI.
- The SPI is not pending.

16-bit accesses to bits [15:0] of this register must be supported.

## Configuration

If [GICD\\_TYPER](#).MBIS == 0, this register is reserved.

When [GICD\\_CTLR](#).DS==1, this register is WI.

## Attributes

GICD\_CLRSPI\_SR is a 32-bit register.

## Field descriptions

The GICD\_CLRSPI\_SR 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	INTID									

### Bits [31:10]

Reserved, RES0.

### INTID, bits [9:0]

The INTID of the SPI.

The function of this register depends on whether the targeted SPI is configured to be an edge-triggered or level-sensitive interrupt:

- For an edge-triggered interrupt, a write to [GICD\\_SETSPI\\_NSR](#) or [GICD\\_SETSPI\\_SR](#) adds the pending state to the targeted interrupt. It will stop being pending on activation, or if the pending state is removed by a write to [GICD\\_CLRSPI\\_NSR](#), [GICD\\_CLRSPI\\_SR](#), or [GICD\\_ICPENDR<n>](#).

- For a level-sensitive interrupt, a write to [GICD\\_SETSPI\\_NSR](#) or [GICD\\_SETSPI\\_SR](#) adds the pending state to the targeted interrupt. It will remain pending until it is deasserted by a write to [GICD\\_CLRSPI\\_NSR](#) or GICD\_CLRSPI\_SR. If the interrupt is activated between having the pending state added and being deactivated, then the interrupt will be active and pending.

## Accessing the GICD\_CLRSPI\_SR

GICD\_CLRSPI\_SR can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	0x0058

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_CPENDSGIR<n>, SGI Clear-Pending Registers, n = 0 - 3

The GICD\_CPENDSGIR<n> characteristics are:

## Purpose

Removes the pending state from an SGI.

A write to this register changes the state of a pending SGI to inactive, and the state of an active and pending SGI to active.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

These registers are used only when affinity routing is not enabled. When affinity routing is enabled, this register is RES0. An implementation is permitted to make the register RAZ/WI in this case.

A register bit that corresponds to an unimplemented SGI is RAZ/WI.

These registers are byte-accessible.

If the GIC implementation supports two Security states:

- A register bit that corresponds to a Group 0 interrupt is RAZ/WI to Non-secure accesses.
- Register bits corresponding to unimplemented PEs are RAZ/WI.

## Configuration

Some or all RW fields of this register have defined reset values.

Four SGI clear-pending registers are implemented. Each register contains eight clear-pending bits for each of four SGIs, for a total of 16 possible SGIs.

In multiprocessor implementations, each PE has a copy of these registers.

## Attributes

GICD\_CPENDSGIR<n> is a 32-bit register.

## Field descriptions

The GICD\_CPENDSGIR<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
SGI_clear_pending_bits<x>, bits [8x+7:8x], for x = 0 to 3																															

### SGI\_clear\_pending\_bits<x>, bits [8x+7:8x], for x = 0 to 3

Removes the pending state from SGI number 4n + x for the PE corresponding to the bit number written to.

Reads and writes have the following behavior:

SGI_clear_pending_bits<x>	Meaning
0	If read, indicates that the SGI from the corresponding PE is not pending and is not active and pending. If written, has no effect.
1	If read, indicates that the SGI from the corresponding PE is pending or is active and pending. If written, removes the pending state from the SGI for the corresponding PE.

When this register has an architecturally-defined reset value, this field resets to 0.

For SGI ID m, generated by processing element C writing to the corresponding [GICD\\_SGIR](#) field, where DIV and MOD are the integer division and modulo operations:

- The corresponding GICD\_CPENDSGIR<n> number is given by  $n = m \text{ DIV } 4$ .
- The offset of the required register is  $(0xF10 + (4n))$ .
- The offset of the required field within the register GICD\_CPENDSGIR<n> is given by  $m \text{ MOD } 4$ .
- The required bit in the 8-bit SGI clear-pending field m is bit C.

## Accessing the GICD\_CPENDSGIR<n>

GICD\_CPENDSGIR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0x0F10 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

## GICD\_CTLR, Distributor Control Register

The GICD\_CTLR characteristics are:

## Purpose

Enables interrupts and affinity routing.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

If an interrupt is pending within a CPU interface when the corresponding GICD\_CTLR.EnableGrpX bit is written from 1 to 0 the interrupt must be retrieved from the CPU interface.

### Note

This might have no effect on the forwarded interrupt if it has already been activated.

When a write changes the value of ARE for a Security state or the value of the DS bit, the format used for interpreting the remaining bits provided in the write data is the format that applied before the write takes effect.

## Configuration

Some or all RW fields of this register have defined reset values.

The format of this register depends on the Security state of the access and the number of Security states supported, which is specified by GICD\_CTLR.DS.

## Attributes

GICD\_CTLR is a 32-bit register.

## Field descriptions

The GICD CTLR bit assignments are:

**When access is Secure, in a system that supports two Security states:**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
RWP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	E1	NW	FS	ARE	NS	ARE	S0	EnableGrp1S	EnableGrp1NS	EnableGrp0

**RWP, bit [31]**

Register Write Pending. Read only. Indicates whether a register write is in progress or not.

RWP	Meaning
0	No register write in progress. The effects of previous register writes to the affected register fields are visible to all logical components of the GIC architecture, including the CPU interfaces.
1	Register write in progress. The effects of previous register writes to the affected register fields are not guaranteed to be visible to all logical components of the GIC architecture, including the CPU interfaces, as the effects of the changes are still being propagated.

This field tracks writes to:

- GICD\_CTLR[2:0], the Group Enables, for transitions from 1 to 0 only.
- GICD\_CTLR[7:4], the ARE bits, EINWF bit and DS bit.
- GICD\_ICENABLER<n>.

Updates to other register fields are not tracked by this field.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### Bits [30:8]

Reserved, RES0.

### E1NWF, bit [7]

Enable 1 of N Wakeup Functionality.

It is IMPLEMENTATION DEFINED whether this bit is programmable, or RAZ/WI.

If it is implemented, then it has the following behavior:

E1NWF	Meaning
0	A PE that is asleep cannot be picked for 1 of N interrupts.
1	A PE that is asleep can be picked for 1 of N interrupts as determined by IMPLEMENTATION DEFINED controls.

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.

### DS, bit [6]

Disable Security.

DS	Meaning
0	Non-secure accesses are not permitted to access and modify registers that control Group 0 interrupts.
1	Non-secure accesses are permitted to access and modify registers that control Group 0 interrupts.

If DS is written from 0 to 1 when GICD\_CTLR.ARE\_S == 1, then GICD\_CTLR.ARE for the single Security state is RAO/WI.

If the Distributor only supports a single Security state, this bit is RAO/WI.

If the Distributor supports two Security states, it IMPLEMENTATION DEFINED whether this bit is programmable or implemented as RAZ/WI.

When this field is set to 1, all accesses to GICD\_CTLR access the single Security state view, and all bits are accessible.

When set to 1, this field can only be cleared by a hardware reset.

Writing this bit from 0 to 1 is UNPREDICTABLE if any of the following is true:

- [GICD\\_CTLR.EnableGrp0](#)==1.
- [GICD\\_CTLR.EnableGrp1S](#)==1.
- [GICD\\_CTLR.EnableGrp1NS](#)==1.
- One or more INTID is in the Active or Active and Pending state.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**ARE\_NS, bit [5]**

Affinity Routing Enable, Non-secure state.

ARE_NS	Meaning
0	Affinity routing disabled for Non-secure state.
1	Affinity routing enabled for Non-secure state.

When affinity routing is enabled for the Secure state, this field is RAO/WI.

Changing the ARE\_NS settings from 0 to 1 is UNPREDICTABLE except when GICD\_CTLR.EnableGrp1 Non-Secure == 0.

Changing the ARE\_NS settings from 1 to 0 is UNPREDICTABLE.

If GICv2 backwards compatibility for Non-secure state is not implemented, this field is RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**ARE\_S, bit [4]**

Affinity Routing Enable, Secure state.

ARE_S	Meaning
0	Affinity routing disabled for Secure state.
1	Affinity routing enabled for Secure state.

Changing the ARE\_S setting from 0 to 1 is UNPREDICTABLE except when all of the following apply:

- GICD\_CTLR.EnableGrp0==0.
- GICD\_CTLR.EnableGrp1S==0.
- GICD\_CTLR.EnableGrp1NS==0.

Changing the ARE\_S settings from 1 to 0 is UNPREDICTABLE.

If GICv2 backwards compatibility for Secure state is not implemented, this field is RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**Bit [3]**

Reserved, RES0.

**EnableGrp1S, bit [2]**

Enable Secure Group 1 interrupts.

EnableGrp1S	Meaning
0	Secure Group 1 interrupts are disabled.
1	Secure Group 1 interrupts are enabled.

If GICD\_CTLR.ARE\_S == 0, this field is RES0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**EnableGrp1NS, bit [1]**

Enable Non-secure Group 1 interrupts.

EnableGrp1NS	Meaning
0	Non-secure Group 1 interrupts are disabled.
1	Non-secure Group 1 interrupts are enabled.

**Note**

This field also controls whether LPIs are forwarded to the PE.



When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### EnableGrp0, bit [0]

Enable Group 0 interrupts.

EnableGrp0	Meaning
0	Group 0 interrupts are disabled.
1	Group 0 interrupts are enabled.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## When access is Non-secure, in a system that supports two Security states:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RWP	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	ARE_NS	0	0	EnableGrp1A	EnableGrp1	

### RWP, bit [31]

This bit is a read-only alias of the Secure GICD\_CTLR.RWP bit.

### Bits [30:5]

Reserved, RES0.

### ARE\_NS, bit [4]

This bit is a read-write alias of the Secure GICD\_CTLR.ARE\_NS bit.

If GICv2 backwards compatibility for Non-secure state is not implemented, this field is RAO/WI.

### Bits [3:2]

Reserved, RES0.

### EnableGrp1A, bit [1]

If ARE\_NS == 1, then this bit is a read-write alias of the Secure GICD\_CTLR.EnableGrp1NS bit.

If ARE\_NS == 0, then this bit is RES0.

### EnableGrp1, bit [0]

If ARE\_NS == 0, then this bit is a read-write alias of the Secure GICD\_CTLR.EnableGrp1NS bit.

If ARE\_NS == 1, then this bit is RES0.

## When in a system that supports only a single Security state:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RWP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	E1NWFDS	0	ARE	0	0	EnableGrp1	EnableGrp0	

### RWP, bit [31]

Register Write Pending. Read only. Indicates whether a register write is in progress or not:

RWP	Meaning
0	No register write in progress. The effects of previous register writes to the affected register fields are visible to all logical components of the GIC architecture, including the CPU interfaces.
1	Register write in progress. The effects of previous register writes to the affected register fields are not guaranteed to be visible to all logical components of the GIC architecture, including the CPU interfaces, as the effects of the changes are still being propagated.

This field tracks updates to:

- GICD\_CTLR[2:0], the Group Enables, for transitions from 1 to 0 only.
- GICD\_CTLR[7:4], the ARE bits, E1NWF bit and DS bit.
- GICD\_ICENABLER<n>, the bits that allow disabling of SPIs.

Updates to other register fields are not tracked by this field.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### Bits [30:8]

Reserved, RES0.

### E1NWF, bit [7]

Enable 1 of N Wakeup Functionality.

It is IMPLEMENTATION DEFINED whether this bit is programmable, or RAZ/WI.

If it is implemented, then it has the following behavior:

E1NWF	Meaning
0	A PE that is asleep cannot be picked for 1 of N interrupts.
1	A PE that is asleep can be picked for 1 of N interrupts as determined by IMPLEMENTATION DEFINED controls.

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.

### DS, bit [6]

Disable Security. This field is RAO/WI.

### Bit [5]

Reserved, RES0.

### ARE, bit [4]

Affinity Routing Enable.

ARE	Meaning
0	Affinity routing disabled.
1	Affinity routing enabled.

Changing the ARE settings from 0 to 1 is UNPREDICTABLE except when all of the following apply:

- GICD\_CTLR.EnableGrp1==0.
- GICD\_CTLR.EnableGrp0==0.

Changing ARE from 1 to 0 is UNPREDICTABLE.

If GICv2 backwards compatibility is not implemented, this field is RAO/WI.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**Bits [3:2]**

Reserved, RES0.

**EnableGrp1, bit [1]**

Enable Group 1 interrupts.

EnableGrp1	Meaning
0	Group 1 interrupts disabled.
1	Group 1 interrupts enabled.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**EnableGrp0, bit [0]**

Enable Group 0 interrupts.

EnableGrp0	Meaning
0	Group 0 interrupts are disabled.
1	Group 0 interrupts are enabled.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Accessing the GICD\_CTLR**

GICD\_CTLR can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	0x0000

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_ICACTIVER<n>, Interrupt Clear-Active Registers, n = 0 - 31

The GICD\_ICACTIVER<n> characteristics are:

## Purpose

Deactivates the corresponding interrupt. These registers are used when saving and restoring GIC state.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

When affinity routing is enabled for the Security state of an interrupt, the bits corresponding to SGIs and PPIs in that Security state are RAZ/WI, and equivalent functionality for SGIs and PPIs is provided by [GICR\\_ICACTIVER0](#).

Bits corresponding to unimplemented interrupts are RAZ/WI.

If [GICD\\_CTLR.DS](#)==0, unless the [GICD\\_NSACR<n>](#) registers permit Non-secure software to control Group 0 and Secure Group 1 interrupts, any bits that correspond to Group 0 or Secure Group 1 interrupts are accessible only by Secure accesses and are RAZ/WI to Non-secure accesses.

## Configuration

Some or all RW fields of this register have defined reset values.

These registers are available in all GIC configurations. If [GICD\\_CTLR.DS](#)==0, these registers are Common.

The number of implemented GICD\_ICACTIVER<n> registers is ([GICD\\_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD\_ICACTIVER0 is Banked for each connected PE with [GICR\\_TYPER.Processor\\_Number](#) < 8.

Accessing GICD\_ICACTIVER0 from a PE with [GICR\\_TYPER.Processor\\_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

## Attributes

GICD\_ICACTIVER<n> is a 32-bit register.

## Field descriptions

The GICD\_ICACTIVER<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
Clear_active_bit<x>, bit [x], for x = 0 to 31																															

### Clear\_active\_bit<x>, bit [x], for x = 0 to 31

Removes the active state from interrupt number 32n + x. Reads and writes have the following behavior:

Clear_active_bit<x>	Meaning
0	If read, indicates that the corresponding interrupt is not active, and is not active and pending. If written, has no effect.
1	If read, indicates that the corresponding interrupt is active, or is active and pending. If written, deactivates the corresponding interrupt, if the interrupt is active. If the interrupt is already deactivated, the write has no effect.

When this register has an architecturally-defined reset value, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD\_ICACTIVER<n> number, n, is given by  $n = m \text{ DIV } 32$ .
- The offset of the required GICD\_ICACTIVER is  $(0 \times 380 + (4 * n))$ .
- The bit number of the required group modifier bit in this register is  $m \text{ MOD } 32$ .

## Accessing the GICD\_ICACTIVER<n>

GICD\_ICACTIVER<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 0380 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_ICENABLER<n>, Interrupt Clear-Enable Registers, n = 0 - 31

The GICD\_ICENABLER<n> characteristics are:

## Purpose

Disables forwarding of the corresponding interrupt to the CPU interfaces.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

For SGIs and PPIs:

- When ARE is 1 for the Security state of an interrupt, the field for that interrupt is RES0 and an implementation is permitted to make the field RAZ/WI in this case.
- Equivalent functionality is provided by GICR\_ICENABLER0.

Bits corresponding to unimplemented interrupts are RAZ/WI.

When [GICD\\_CTLR.DS](#)==0, bits corresponding to Group 0 and Secure Group 1 interrupts are RAZ/WI to Non-secure accesses.

It is IMPLEMENTATION DEFINED whether implemented SGIs are permanently enabled, or can be enabled and disabled by writes to [GICD\\_ISENABLER<n>](#) and [GICD\\_ICENABLER<n>](#) where n=0.

Completion of a write to this register does not guarantee that the effects of the write are visible throughout the affinity hierarchy. To ensure an enable has been cleared, software must write to the register with bits set to 1 to clear the required enables. Software must then poll [GICD\\_CTLR.RWP](#) until it has the value zero.

## Configuration

Some or all RW fields of this register have defined reset values.

These registers are available in all GIC configurations. If [GICD\\_CTLR.DS](#)==0, these registers are Common.

The number of implemented [GICD\\_ICENABLER<n>](#) registers is ([GICD\\_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD\_ICENABLER0 is Banked for each connected PE with [GICR\\_TYPER.Processor\\_Number](#) < 8.

Accessing GICD\_ICENABLER0 from a PE with [GICR\\_TYPER.Processor\\_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

## Attributes

GICD\_ICENABLER<n> is a 32-bit register.

## Field descriptions

The GICD\_ICENABLER<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
Clear_enable_bit<x>, bit [x], for x = 0 to 31																															

**Clear\_enable\_bit<x>, bit [x], for x = 0 to 31**

For SPIs and PPIs, controls the forwarding of interrupt number  $32n + x$  to the CPU interfaces. Reads and writes have the following behavior:

Clear_enable_bit<x>	Meaning
0	If read, indicates that forwarding of the corresponding interrupt is disabled. If written, has no effect.
1	If read, indicates that forwarding of the corresponding interrupt is enabled. If written, disables forwarding of the corresponding interrupt. After a write of 1 to this bit, a subsequent read of this bit returns 0.

For SGIs, the behavior of this bit is IMPLEMENTATION DEFINED.

When this register has an architecturally-defined reset value, this field resets to an architecturally UNKNOWN value.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD\_ICENABLER<n> number, n, is given by  $n = m \text{ DIV } 32$ .
- The offset of the required GICD\_ICENABLER is  $(0 \times 180 + (4 * n))$ .
- The bit number of the required group modifier bit in this register is  $m \text{ MOD } 32$ .

**Note**

Writing a 1 to a GICD\_ICENABLER<n> bit only disables the forwarding of the corresponding interrupt from the Distributor to any CPU interface. It does not prevent the interrupt from changing state, for example becoming pending or active and pending if it is already active.

**Accessing the GICD\_ICENABLER<n>**

GICD\_ICENABLER<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 0180 + 4n$

# GICD\_ICFGR<n>, Interrupt Configuration Registers, n = 0 - 63

The GICD\_ICFGR<n> characteristics are:

## Purpose

Determines whether the corresponding interrupt is edge-triggered or level-sensitive.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

## Configuration

Some or all RW fields of this register have defined reset values.

These registers are available in all GIC configurations. If the GIC implementation supports two Security states, these registers are Common.

GICD\_ICFGR1 is Banked for each connected PE with [GICR\\_TYPER](#).Processor\_Number < 8.

Accessing GICD\_ICFGR1 from a PE with [GICR\\_TYPER](#).Processor\_Number > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

For SGIs and PPIs:

- When ARE is 1 for the Security state of an interrupt, the field for that interrupt is RES0 and an implementation is permitted to make the field RAZ/WI in this case.
- Equivalent functionality is provided by GICR\_ICFGR<n>.

For each supported PPI, it is IMPLEMENTATION DEFINED whether software can program the corresponding Int\_config field.

For SGIs, Int\_config fields are RO, meaning that GICD\_ICFGR0 is RO.

Software must disable an interrupt before the value of the corresponding programmable Int\_config field is changed. GIC behavior is otherwise UNPREDICTABLE.

Changing the interrupt configuration between level-sensitive and edge-triggered (in either direction) at a time when there is a pending interrupt will leave the interrupt in an UNKNOWN pending state.

Fields corresponding to unimplemented interrupts are RAZ/WI.

## Attributes

GICD\_ICFGR<n> is a 32-bit register.

## Field descriptions

The GICD\_ICFGR<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
<a href="#">Int_config&lt;x&gt;, bits [2x+1:2x], for x = 0 to 15</a>																															



**Int\_config<x>, bits [2x+1:2x], for x = 0 to 15**

Indicates whether the interrupt with ID  $16n + x$  is level-sensitive or edge-triggered.

Int\_config[0] (bit [2x]) is RES0.

Possible values of Int\_config[1] (bit [2x+1]) are:

Int_config<x>	Meaning
0	Corresponding interrupt is level-sensitive.
1	Corresponding interrupt is edge-triggered.

For SGIs, Int\_config[1] is RAO/WI.

For SPIs and PPIs, Int\_config[1] is programmable unless the implementation supports two Security states and the bit corresponds to a Group 0 or Secure Group 1 interrupt, in which case the bit is RAZ/WI to Non-secure accesses.

When this register has an architecturally-defined reset value, this field resets to an architecturally UNKNOWN value.

**Accessing the GICD\_ICFGR<n>**

GICD\_ICFGR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 0C00 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_ICPENDR<n>, Interrupt Clear-Pending Registers, n = 0 - 31

The GICD\_ICPENDR<n> characteristics are:

## Purpose

Removes the pending state from the corresponding interrupt.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

Clear-pending bits for SGIs are RO/WI.

When affinity routing is enabled for the Security state of an interrupt:

- Bits corresponding to SGIs and PPIs are RAZ/WI, and equivalent functionality for SGIs and PPIs is provided by [GICR\\_ICPENDR0](#).
- Bits corresponding to Group 0 and Group 1 Secure interrupts can only be cleared by Secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

If [GICD\\_CTLR](#).DS==0, unless the [GICD\\_NSACR<n>](#) registers permit Non-secure software to control Group 0 and Secure Group 1 interrupts, any bits that correspond to Group 0 or Secure Group 1 interrupts are accessible only by Secure accesses and are RAZ/WI to Non-secure accesses.

## Configuration

Some or all RW fields of this register have defined reset values.

These registers are available in all GIC configurations. If [GICD\\_CTLR](#).DS==0, these registers are Common.

The number of implemented [GICD\\_ICPENDR<n>](#) registers is ([GICD\\_TYPER](#).ITLinesNumber+1). Registers are numbered from 0.

GICD\_ICPENDR0 is Banked for each connected PE with [GICR\\_TYPER](#).Processor\_Number < 8.

Accessing GICD\_ICPENDR0 from a PE with [GICR\\_TYPER](#).Processor\_Number > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

## Attributes

GICD\_ICPENDR<n> is a 32-bit register.

## Field descriptions

The GICD\_ICPENDR<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
Clear_pending_bit<x>, bit [x], for x = 0 to 31																															

**Clear\_pending\_bit<x>, bit [x], for x = 0 to 31**

For SPIs and PPIs, removes the pending state from interrupt number  $32n + x$ . Reads and writes have the following behavior:

Clear_pending_bit<x>	Meaning
0	If read, indicates that the corresponding interrupt is not pending on any PE. If written, has no effect.
1	If read, indicates that the corresponding interrupt is pending, or active and pending: <ul style="list-style-type: none"> <li>On this PE if the interrupt is an SGI or PPI.</li> <li>On at least one PE if the interrupt is an SPI.</li> </ul> If written, changes the state of the corresponding interrupt from pending to inactive, or from active and pending to active. This has no effect in the following cases: <ul style="list-style-type: none"> <li>If the interrupt is an SGI. In this case, the write is ignored. The pending state of an SGI can be cleared using <a href="#">GICD_CPENDSGIR&lt;n&gt;</a>.</li> <li>If the interrupt is not pending and is not active and pending.</li> <li>If the interrupt is a level-sensitive interrupt that is pending or active and pending for a reason other than a write to <a href="#">GICD_ISPENDR&lt;n&gt;</a>. In this case, if the interrupt signal continues to be asserted, the interrupt remains pending or active and pending.</li> </ul>

When this register has an architecturally-defined reset value, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD\_ICPENDR<n> number, n, is given by  $n = m \text{ DIV } 32$ .
- The offset of the required GICD\_ICPENDR is  $(0 \times 200 + (4 * n))$ .
- The bit number of the required group modifier bit in this register is  $m \text{ MOD } 32$ .

**Accessing the GICD\_ICPENDR<n>**

GICD\_ICPENDR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 0280 + 4n$

# GICD\_IGROUPR<n>, Interrupt Group Registers, n = 0 - 31

The GICD\_IGROUPR<n> characteristics are:

## Purpose

Controls whether the corresponding interrupt is in Group 0 or Group 1.  
This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RAZ/WI

For SGIs and PPIs:

- When ARE is 1 for the Security state of an interrupt, the field for that interrupt is RES0 and an implementation is permitted to make the field RAZ/WI in this case.
- Equivalent functionality is provided by GICR\_IGROUPR0.

When [GICD\\_CTLR.DS](#)==0, the register is RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

---

### Note

Accesses to GICD\_IGROUPR0 when affinity routing is not enabled for a Security state access the same state as [GICR\\_IGROUPR0](#), and must update Redistributor state associated with the PE performing the accesses.

Implementations must ensure that an interrupt that is pending at the time of the write uses either the old value or the new value and must ensure that the interrupt is neither lost nor handled more than once. The effect of the change must be visible in finite time.

---

## Configuration

GICD\_IGROUPR0 resets to an IMPLEMENTATION DEFINED value, that might be UNKNOWN.

GICD\_IGROUPR<n> where n is greater than 0 resets to 0x00000000.

These registers are available in all GIC configurations. If [GICD\\_CTLR.DS](#)==0, these registers are Secure.

The number of implemented GICD\_IGROUPR<n> registers is ([GICD\\_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD\_IGROUPR0 is Banked for each connected PE with [GICR\\_TYPER.Processor\\_Number](#) < 8.

Accessing GICD\_IGROUPR0 from a PE with [GICR\\_TYPER.Processor\\_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

## Attributes

GICD\_IGROUPR<n> is a 32-bit register.

## Field descriptions

The GICD\_IGROUPR<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
Group_status_bit<x>, bit [x], for x = 0 to 31																															

### Group\_status\_bit<x>, bit [x], for x = 0 to 31

Group status bit.

Group_status_bit<x>	Meaning
0	When <a href="#">GICD_CTLR.DS</a> ==1, the corresponding interrupt is Group 0.
1	When <a href="#">GICD_CTLR.DS</a> ==0, the corresponding interrupt is Secure. When <a href="#">GICD_CTLR.DS</a> ==1, the corresponding interrupt is Group 1. When <a href="#">GICD_CTLR.DS</a> ==0, the corresponding interrupt is Non-secure Group 1.

If affinity routing is enabled for the Security state of an interrupt, the bit that corresponds to the interrupt is concatenated with the equivalent bit in [GICD\\_IGRPMODR<n>](#) to form a 2-bit field that defines an interrupt group. The encoding of this field is described in [GICD\\_IGRPMODR<n>](#).

If affinity routing is disabled for the Security state of an interrupt, then:

- The corresponding [GICD\\_IGRPMODR<n>](#) bit is RES0.
- For Secure interrupts, the interrupt is Secure Group 0.
- For Non-secure interrupts, the interrupt is Non-secure Group 1.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD\_IGROUP<n> number, n, is given by  $n = m \text{ DIV } 32$ .
- The offset of the required GICD\_IGROUP is  $(0 \times 080 + (4 * n))$ .
- The bit number of the required group modifier bit in this register is  $m \text{ MOD } 32$ .

## Accessing the GICD\_IGROUPR<n>

GICD\_IGROUPR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 0080 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_IGRPMODR<n>, Interrupt Group Modifier Registers, n = 0 - 31

The GICD\_IGRPMODR<n> characteristics are:

## Purpose

When [GICD\\_CTLR.DS](#)==0, this register together with the [GICD\\_IGROUPR<n>](#) registers, controls whether the corresponding interrupt is in:

- Secure Group 0.
- Non-secure Group 1.
- Secure Group 1.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RAZ/WI

When affinity routing is enabled for Secure state, GICD\_IGRPMODR0 is RES0 and equivalent functionality is proved by [GICR\\_IGRPMODR0](#).

When [GICD\\_CTLR.DS](#)==0, the register is RAZ/WI to Non-secure accesses.

### Note

Implementations must ensure that an interrupt that is pending at the time of the write uses either the old value or the new value and must ensure that the interrupt is neither lost nor handled more than once. The effect of the change must be visible in finite time.

## Configuration

Some or all RW fields of this register have defined reset values.

When [GICD\\_CTLR.DS](#)==0, these registers are Secure.

The number of implemented [GICD\\_IGROUPR<n>](#) registers is ([GICD\\_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

When [GICD\\_CTLR.ARE\\_S](#)==0 or [GICD\\_CTLR.DS](#)==1, the GICD\_IGRPMODR<n> registers are RES0. An implementation can make these registers RAZ/WI in this case.

## Attributes

GICD\_IGRPMODR<n> is a 32-bit register.

## Field descriptions

The GICD\_IGRPMODR<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
<a href="#">Group_modifier_bit&lt;x&gt;</a> , bit [x], for x = 0 to 31																															

**Group\_modifier\_bit<x>, bit [x], for x = 0 to 31**

Group modifier bit. When affinity routing is enabled for the Security state of an interrupt, the bit that corresponds to the interrupt is concatenated with the equivalent bit in [GICD\\_IGROUPR<n>](#) to form a 2-bit field that defines an interrupt group:

Group modifier bit	Group status bit	Definition	Short name
0	0	Secure Group 0	G0S
0	1	Non-secure Group 1	G1NS
1	0	Secure Group 1	G1S
1	1	Reserved, treated as Non-secure Group 1	-

When this register has an architecturally-defined reset value, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD\_IGRPMODR<n> number, n, is given by  $n = m \text{ DIV } 32$ .
- The offset of the required GICD\_IGRPMODR is  $(0 \times 080 + (4 * n))$ .
- The bit number of the required group modifier bit in this register is  $m \text{ MOD } 32$ .

See [GICD\\_IGROUPR<n>](#) for information about the GICD\_IGRPMODR0 reset value.

**Accessing the GICD\_IGRPMODR<n>**

GICD\_IGRPMODR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 0D00 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_IIDR, Distributor Implementer Identification Register

The GICD\_IIDR characteristics are:

## Purpose

Provides information about the implementer and revision of the Distributor.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

## Configuration

This register is available in all configurations of the GIC. If the GIC implementation supports two Security states, this register is Common.

## Attributes

GICD\_IIDR is a 32-bit register.

## Field descriptions

The GICD\_IIDR 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
ProductID								0	0	0	0	Variant			Revision			Implementer													

### ProductID, bits [31:24]

An IMPLEMENTATION DEFINED product identifier.

### Bits [23:20]

Reserved, RES0.

### Variant, bits [19:16]

An IMPLEMENTATION DEFINED variant number. Typically, this field is used to distinguish product variants, or major revisions of a product.

### Revision, bits [15:12]

An IMPLEMENTATION DEFINED revision number. Typically, this field is used to distinguish minor revisions of a product.

### Implementer, bits [11:0]

Contains the JEP106 code of the company that implemented the Distributor:

- Bits [11:8] are the JEP106 continuation code of the implementer. For an ARM implementation, this field is 0x4.



- Bit [7] is always 0.
- Bits [6:0] are the JEP106 identity code of the implementer. For an ARM implementation, bits [7:0] are therefore 0x3B.

## Accessing the GICD\_IIDR

GICD\_IIDR can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	0x0008

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_IPRIORITYR<n>, Interrupt Priority Registers, n = 0 - 254

The GICD\_IPRIORITYR<n> characteristics are:

## Purpose

Holds the priority of the corresponding interrupt.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

These registers are always used when affinity routing is not enabled. When affinity routing is enabled for the Security state of an interrupt:

- [GICR\\_IPRIORITYR<n>](#) is used instead of GICD\_IPRIORITYR<n> where n = 0 to 7 (that is, for SGIs and PPIs).
- GICD\_IPRIORITYR<n> is RAZ/WI where n = 0 to 7.

These registers are byte-accessible.

A register field corresponding to an unimplemented interrupt is RAZ/WI.

A GIC might implement fewer than eight priority bits, but must implement at least bits [7:4] of each field. In each field, unimplemented bits are RAZ/WI, see Interrupt prioritization.

When [GICD\\_CTLR.DS](#)==0:

- A register bit that corresponds to a Group 0 or Secure Group 1 interrupt is RAZ/WI to Non-secure accesses.
- A Non-secure access to a field that corresponds to a Non-secure Group 1 interrupt behaves as described in Software views of interrupt priority.

It is IMPLEMENTATION DEFINED whether changing the value of a priority field changes the priority of an active interrupt.

---

### Note

Implementations must ensure that an interrupt that is pending at the time of the write uses either the old value or the new value and must ensure that the interrupt is neither lost nor handled more than once. The effect of the change must be visible in finite time.

---

## Configuration

Some or all RW fields of this register have defined reset values.

These registers are available in all configurations of the GIC. When [GICD\\_CTLR.DS](#)==0, these registers are Common.

The number of implemented GICD\_IPRIORITYR<n> registers is 8\*([GICD\\_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD\_IPRIORITYR0 to GICD\_IPRIORITYR7 are Banked for each connected PE with [GICR\\_TYPER.Processor\\_Number](#) < 8.

Accessing GICD\_IPRIORITYR0 to GICD\_IPRIORITYR7 from a PE with [GICR\\_TYPER.Processor\\_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

## Attributes

GICD\_IPRIORITYR<n> is a 32-bit register.

## Field descriptions

The GICD\_IPRIORITYR<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
Priority_offset_3B								Priority_offset_2B								Priority_offset_1B								Priority_offset_0B							

### Priority\_offset\_3B, bits [31:24]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 3. Lower priority values correspond to greater priority of the interrupt.

When this register has an architecturally-defined reset value, this field resets to 0.

### Priority\_offset\_2B, bits [23:16]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 2. Lower priority values correspond to greater priority of the interrupt.

When this register has an architecturally-defined reset value, this field resets to 0.

### Priority\_offset\_1B, bits [15:8]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 1. Lower priority values correspond to greater priority of the interrupt.

When this register has an architecturally-defined reset value, this field resets to 0.

### Priority\_offset\_0B, bits [7:0]

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 0. Lower priority values correspond to greater priority of the interrupt.

When this register has an architecturally-defined reset value, this field resets to 0.

For interrupt ID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD\_IPRIORITYR<n> number, n, is given by  $n = m \text{ DIV } 4$ .
- The offset of the required GICD\_IPRIORITYR<n> register is  $(0 \times 400 + (4 * n))$ .
- The byte offset of the required Priority field in this register is  $m \text{ MOD } 4$ , where:
  - Byte offset 0 refers to register bits [7:0].
  - Byte offset 1 refers to register bits [15:8].
  - Byte offset 2 refers to register bits [23:16].
  - Byte offset 3 refers to register bits [31:24].

## Accessing the GICD\_IPRIORITYR<n>

GICD\_IPRIORITYR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 400 + 4n$

# GICD\_IROUTER<n>, Interrupt Routing Registers, n = 32 - 1019

The GICD\_IROUTER<n> characteristics are:

## Purpose

When affinity routing is enabled, provides routing information for the SPI with INTID n.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

These registers are used only when affinity routing is enabled. When affinity routing is not enabled:

- These registers are RES0. An implementation is permitted to make the register RAZ/WI in this case.
- The [GICD\\_ITARGETSR<n>](#) registers provide interrupt routing information.

### Note

When affinity routing becomes enabled for a Security state (for example, following a reset or following a write to [GICD\\_CTLR](#)) the value of all writeable fields in this register is UNKNOWN for that Security state. When the group of an interrupt changes so the ARE setting for the interrupt changes to 1, the value of this register is UNKNOWN for that interrupt.

If [GICD\\_CTLR](#).DS==0, unless the [GICD\\_NSACR<n>](#) registers permit Non-secure software to control Group 0 and Secure Group 1 interrupts, any GICD\_IROUTER<n> registers that correspond to Group 0 or Secure Group 1 interrupts are accessible only by Secure accesses and are RAZ/WI to Non-secure accesses.

### Note

For each interrupt, a GIC implementation might support fewer than 256 values for an affinity level. In this case, some bits of the corresponding affinity level field might be RO.

Implementations must ensure that an interrupt that is pending at the time of the write uses either the old value or the new value and must ensure that the interrupt is neither lost nor handled more than once. The effect of the change must be visible in finite time.

## Configuration

These registers are available in all configurations of the GIC. If the GIC implementation supports two Security states, these registers are Common.

The maximum value of n is given by  $(32 * (\text{GICD\_TYPER.ITLinesNumber} + 1) - 1)$ . [GICD\\_IROUTER<n>](#) registers where n=0 to 31 are reserved.

## Attributes

GICD\_IROUTER<n> is a 64-bit register.

## Field descriptions

The GICD\_IROUTER<n> 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	Aff3							
Interrupt_Routing_Mode	0	0	0	0	0	0	0	Aff2								Aff1								Aff0							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:40]

Reserved, RES0.

Aff3, bits [39:32]

Affinity level 3, the least significant affinity level field.

Interrupt\_Routing\_Mode, bit [31]

Interrupt Routing Mode. Defines how SPIs are routed in an affinity hierarchy:

Interrupt_Routing_Mode	Meaning
0	Interrupts routed to the PE specified by a.b.c.d. In this routing, a, b, c, and d are the values of fields Aff3, Aff2, Aff1, and Aff0 respectively.
1	Interrupts routed to any PE defined as a participating node.

If GICD\_IROUTER<n>.IRM == 0 and the affinity path does not correspond to an implemented PE, then if the corresponding interrupt becomes pending it will not be forwarded to any PE and will remain pending.

In implementations that do not require 1 of N distribution of SPIs, this bit might be RAZ/WI.

When this bit is set to 1, GICD\_IROUTER<n>.{Aff3, Aff2, Aff1, Aff0} are UNKNOWN.

Note
An implementation might choose to make the Aff<n> fields RO when this field is 1.

Bits [30:24]

Reserved, RES0.

Aff2, bits [23:16]

Affinity level 2, an intermediate affinity level field.

Aff1, bits [15:8]

Affinity level 1, an intermediate affinity level field.

Aff0, bits [7:0]

Affinity level 0, the most significant affinity level field.

For an SPI with INTID m:

- The corresponding GICD\_IROUTER<n> register number, n, is given by n = m.
- The offset of the GICD\_IROUTER<n> register is 0x6000 + 8n.

Accessing the GICD\_IROUTER<n>

GICD\_IROUTER<n> can be accessed through its memory-mapped interface:

Component	Offset
-----------	--------

GIC Distributor	$0 \times 6000 + 8n$
-----------------	----------------------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_ISACTIVER<n>, Interrupt Set-Active Registers, n = 0 - 31

The GICD\_ISACTIVER<n> characteristics are:

## Purpose

Activates the corresponding interrupt. These registers are used when saving and restoring GIC state.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

When affinity routing is enabled for the Security state of an interrupt, bits corresponding to SGIs and PPIs are RAZ/WI, and equivalent functionality for SGIs and PPIs is provided by [GICR\\_ISACTIVER0](#).

Bits corresponding to unimplemented interrupts are RAZ/WI.

If [GICD\\_CTLR.DS](#)==0, unless the [GICD\\_NSACR<n>](#) registers permit Non-secure software to control Group 0 and Secure Group 1 interrupts, any bits that correspond to Group 0 or Secure Group 1 interrupts are accessible only by Secure accesses and are RAZ/WI to Non-secure accesses.

The bit reads as one if the status of the interrupt is active or active and pending. [GICD\\_ISPENDR<n>](#) and [GICD\\_ICPENDR<n>](#) provide the pending status of the interrupt.

## Configuration

Some or all RW fields of this register have defined reset values.

These registers are available in all GIC configurations. If [GICD\\_CTLR.DS](#)==0, these registers are Common.

The number of implemented [GICD\\_ISACTIVER<n>](#) registers is ([GICD\\_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD\_ISACTIVER0 is Banked for each connected PE with [GICR\\_TYPER.Processor\\_Number](#) < 8.

Accessing GICD\_ISACTIVER0 from a PE with [GICR\\_TYPER.Processor\\_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

## Attributes

GICD\_ISACTIVER<n> is a 32-bit register.

## Field descriptions

The GICD\_ISACTIVER<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
<a href="#">Set_active_bit&lt;x&gt;, bit [x], for x = 0 to 31</a>																															

**Set\_active\_bit<x>, bit [x], for x = 0 to 31**

Adds the active state to interrupt number 32n + x. Reads and writes have the following behavior:

Set_active_bit<x>	Meaning
0	If read, indicates that the corresponding interrupt is not active, and is not active and pending. If written, has no effect.
1	If read, indicates that the corresponding interrupt is active, or is active and pending. If written, activates the corresponding interrupt, if the interrupt is not already active. If the interrupt is already active, the write has no effect. After a write of 1 to this bit, a subsequent read of this bit returns 1.

When this register has an architecturally-defined reset value, this field resets to 0.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD\_ISACTIVER<n> number, n, is given by  $n = m \text{ DIV } 32$ .
- The offset of the required GICD\_ISACTIVER is  $(0 \times 300 + (4 * n))$ .
- The bit number of the required group modifier bit in this register is  $m \text{ MOD } 32$ .

## Accessing the GICD\_ISACTIVER<n>

GICD\_ISACTIVER<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 300 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# GICD\_ISENABLER<n>, Interrupt Set-Enable Registers, n = 0 - 31

The GICD\_ISENABLER<n> characteristics are:

## Purpose

Enables forwarding of the corresponding interrupt to the CPU interfaces.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

For SGIs and PPIs:

- When ARE is 1 for the Security state of an interrupt, the field for that interrupt is RES0 and an implementation is permitted to make the field RAZ/WI in this case.
- Equivalent functionality is provided by GICR\_ISENABLER0.

Bits corresponding to unimplemented interrupts are RAZ/WI.

When [GICD\\_CTLR.DS](#)==0, bits corresponding to Group 0 or Secure Group 1 interrupts are RAZ/WI to Non-secure accesses.

It is IMPLEMENTATION DEFINED whether implemented SGIs are permanently enabled, or can be enabled and disabled by writes to [GICD\\_ISENABLER<n>](#) and [GICD\\_ICENABLER<n>](#) where n=0.

For SPIs and PPIs, each bit controls the forwarding of the corresponding interrupt from the Distributor to the CPU interfaces.

## Configuration

Some or all RW fields of this register have defined reset values.

These registers are available in all GIC configurations. If [GICD\\_CTLR.DS](#)==0, these registers are Common.

The number of implemented GICD\_ISENABLER<n> registers is ([GICD\\_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD\_ISENABLER0 is Banked for each connected PE with [GICR\\_TYPER.Processor\\_Number](#) < 8.

Accessing GICD\_ISENABLER0 from a PE with [GICR\\_TYPER.Processor\\_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

## Attributes

GICD\_ISENABLER<n> is a 32-bit register.

## Field descriptions

The GICD\_ISENABLER<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
Set_enable_bit<x>, bit [x], for x = 0 to 31																															

**Set\_enable\_bit<x>, bit [x], for x = 0 to 31**

For SPIs and PPIs, controls the forwarding of interrupt number  $32n + x$  to the CPU interfaces. Reads and writes have the following behavior:

Set_enable_bit<x>	Meaning
0	If read, indicates that forwarding of the corresponding interrupt is disabled. If written, has no effect.
1	If read, indicates that forwarding of the corresponding interrupt is enabled. If written, enables forwarding of the corresponding interrupt. After a write of 1 to this bit, a subsequent read of this bit returns 1.

For SGIs, the behavior of this bit is IMPLEMENTATION DEFINED.

When this register has an architecturally-defined reset value, this field resets to an architecturally UNKNOWN value.

For INTID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD\_ISENABLER<n> number, n, is given by  $n = m \text{ DIV } 32$ .
- The offset of the required GICD\_ISENABLER is  $(0 \times 100 + (4 * n))$ .
- The bit number of the required group modifier bit in this register is  $m \text{ MOD } 32$ .

At start-up, and after a reset, a PE can use this register to discover which peripheral INTIDs the GIC supports. If [GICD\\_CTLR.DS](#)==0 in a system that supports EL3, the PE must do this for the Secure view of the available interrupts, and Non-secure software running on the PE must do this discovery after the Secure software has configured interrupts as Group 0/Secure Group 1 and Non-secure Group 1.

**Accessing the GICD\_ISENABLER<n>**

GICD\_ISENABLER<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 0100 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_ISPENDR<n>, Interrupt Set-Pending Registers, n = 0 - 31

The GICD\_ISPENDR<n> characteristics are:

## Purpose

Adds the pending state to the corresponding interrupt.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

Set-pending bits for SGIs are read-only and ignore writes. The Set-pending bits for SGIs are provided as [GICD\\_SPENDSGIR<n>](#).

When affinity routing is enabled for the Security state of an interrupt:

- Bits corresponding to SGIs and PPIs are RAZ/WI, and equivalent functionality for SGIs and PPIs is provided by GICR\_ISPENDR0.
- Bits corresponding to Group 0 and Group 1 Secure interrupts can only be set by Secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

If [GICD\\_CTLR.DS](#)==0, unless the [GICD\\_NSACR<n>](#) registers permit Non-secure software to control Group 0 and Secure Group 1 interrupts, any bits that correspond to Group 0 or Secure Group 1 interrupts are accessible only by Secure accesses and are RAZ/WI to Non-secure accesses.

## Configuration

Some or all RW fields of this register have defined reset values.

These registers are available in all GIC configurations. If [GICD\\_CTLR.DS](#)==0, these registers are Common.

The number of implemented [GICD\\_ISPENDR<n>](#) registers is ([GICD\\_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD\_ISPENDR0 is Banked for each connected PE with [GICR\\_TYPER.Processor\\_Number](#) < 8.

Accessing GICD\_ISPENDR0 from a PE with [GICR\\_TYPER.Processor\\_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

## Attributes

GICD\_ISPENDR<n> is a 32-bit register.

## Field descriptions

The GICD\_ISPENDR<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
Set_pending_bit<x>, bit [x], for x = 0 to 31																															

**Set\_pending\_bit<x>, bit [x], for x = 0 to 31**

For SPIs and PPIs, adds the pending state to interrupt number  $32n + x$ . Reads and writes have the following behavior:

Set_pending_bit<x>	Meaning
0	If read, indicates that the corresponding interrupt is not pending on any PE. If written, has no effect.
1	If read, indicates that the corresponding interrupt is pending, or active and pending: <ul style="list-style-type: none"> <li>On this PE if the interrupt is an SGI or PPI.</li> <li>On at least one PE if the interrupt is an SPI.</li> </ul> If written, changes the state of the corresponding interrupt from inactive to pending, or from active to active and pending. This has no effect in the following cases: <ul style="list-style-type: none"> <li>If the interrupt is an SGI. The pending state of an SGI can be set using <a href="#">GICD_SPENDSGIR&lt;n&gt;</a>.</li> <li>If the interrupt is not inactive and is not active.</li> <li>If the interrupt is already pending because of a write to <a href="#">GICD_ISPENDR&lt;n&gt;</a>.</li> <li>If the interrupt is already pending because the corresponding interrupt signal is asserted. In this case, the interrupt remains pending if the interrupt signal is deasserted.</li> </ul>

When this register has an architecturally-defined reset value, this field resets to 0.

**Accessing the GICD\_ISPENDR<n>**

GICD\_ISPENDR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 0200 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_ITARGETSR<n>, Interrupt Processor Targets Registers, n = 0 - 254

The GICD\_ITARGETSR<n> characteristics are:

## Purpose

When affinity routing is not enabled, holds the list of target PEs for the interrupt. That is, it holds the list of CPU interfaces to which the Distributor forwards the interrupt if it is asserted and has sufficient priority.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

These registers are used when affinity routing is not enabled. When affinity routing is enabled for the Security state of an interrupt, the target PEs for an interrupt are defined by [GICD\\_IROUTER<n>](#) and the associated byte in GICD\_ITARGETSR<n> is RES0. An implementation is permitted to make the byte RAZ/WI in this case.

- These registers are byte-accessible.
- A register field corresponding to an unimplemented interrupt is RAZ/WI.
- A field bit corresponding to an unimplemented CPU interface is RAZ/WI.
- GICD\_ITARGETSR0-GICD\_ITARGETSR7 are read-only. Each field returns a value that corresponds only to the PE reading the register.
- It is IMPLEMENTATION DEFINED which, if any, SPIs are statically configured in hardware. The field for such an SPI is read-only, and returns a value that indicates the PE targets for the interrupt.
- If [GICD\\_CTLR.DS](#)=0, unless the [GICD\\_NSACR<n>](#) registers permit Non-secure software to control Group 0 and Secure Group 1 interrupts, any bits that correspond to Group 0 or Secure Group 1 interrupts are accessible only by Secure accesses and are RAZ/WI to Non-secure accesses.

In a single connected PE implementation, all interrupts target one PE, and these registers are RAZ/WI.

---

### Note

Implementations must ensure that an interrupt that is pending at the time of the write uses either the old value or the new value and must ensure that the interrupt is neither lost nor handled more than once. The effect of the change must be visible in finite time.

---

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

These registers are available in all configurations of the GIC. When [GICD\\_CTLR.DS](#)=0, these registers are Common.

The number of implemented GICD\_ITARGETSR<n> registers is 8\*([GICD\\_TYPER.ITLinesNumber](#)+1). Registers are numbered from 0.

GICD\_ITARGETSR0 to GICD\_ITARGETSR7 are Banked for each connected PE with [GICR\\_TYPER.Processor\\_Number](#) < 8.

Accessing GICD\_ITARGETSR0 to GICD\_ITARGETSR7 from a PE with [GICR\\_TYPER.Processor\\_Number](#) > 7 is CONSTRAINED UNPREDICTABLE:

- Register is RAZ/WI.
- An UNKNOWN banked copy of the register is accessed.

## Attributes

GICD\_ITARGETSR<n> is a 32-bit register.

## Field descriptions

The GICD\_ITARGETSR<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
CPU_targets_offset_3B								CPU_targets_offset_2B								CPU_targets_offset_1B								CPU_targets_offset_0B							

PEs in the system number from 0, and each bit in a PE targets field refers to the corresponding PE. For example, a value of  $0 \times 3$  means that the Pending interrupt is sent to PEs 0 and 1. For GICD\_ITARGETSR0-GICD\_ITARGETSR7, a read of any targets field returns the number of the PE performing the read.

### CPU\_targets\_offset\_3B, bits [31:24]

PE targets for an interrupt, at byte offset 3.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### CPU\_targets\_offset\_2B, bits [23:16]

PE targets for an interrupt, at byte offset 2.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### CPU\_targets\_offset\_1B, bits [15:8]

PE targets for an interrupt, at byte offset 1.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### CPU\_targets\_offset\_0B, bits [7:0]

PE targets for an interrupt, at byte offset 0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

The bits that are set to 1 in the PE targets field determine which PEs are targeted:

Value of PE targets field	Interrupt targets
0bxxxxxxx1	CPU interface 0
0bxxxxxx1x	CPU interface 1
0bxxxx1xx	CPU interface 2
0bxxxx1xxx	CPU interface 3
0bxxx1xxxx	CPU interface 4
0bxx1xxxxx	CPU interface 5
0bx1xxxxxx	CPU interface 6
0b1xxxxxxx	CPU interface 7

For interrupt ID  $m$ , when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD\_ITARGETSR<n> number,  $n$ , is given by  $n = m \text{ DIV } 4$ .
- The offset of the required GICD\_ITARGETSR<n> register is  $(0 \times 800 + (4 * n))$ .
- The byte offset of the required Priority field in this register is  $m \text{ MOD } 4$ , where:
  - Byte offset 0 refers to register bits [7:0].
  - Byte offset 1 refers to register bits [15:8].
  - Byte offset 2 refers to register bits [23:16].
  - Byte offset 3 refers to register bits [31:24].

Software can write to these registers at any time. Any change to a targets field value:

- Has no effect on any active interrupt. This means that removing a CPU interface from a targets list does not cancel an active state for interrupts on that CPU interface. There is no effect on interrupts that are active and pending until the active status is cleared, at which time it is treated as a pending interrupt.
- Has an effect on any pending interrupts. This means:
  - Enables the CPU interface to be chosen as a target for the pending interrupt using an IMPLEMENTATION DEFINED mechanism.
  - Removing a CPU interface from the target list of a pending interrupt removes the pending state of the interrupt on that CPU interface.

## Accessing the GICD\_ITARGETSR<n>

GICD\_ITARGETSR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 0800 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_NSACR<n>, Non-secure Access Control Registers, n = 0 - 63

The GICD\_NSACR<n> characteristics are:

## Purpose

Enables Secure software to permit Non-secure software on a particular PE to create and control Group 0 interrupts.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RAZ/WI	RW	RAZ/WI

When [GICD\\_CTLR.DS](#)==1, this register is RAZ/WI.

These registers are Secure, and are RAZ/WI to Non-secure accesses.

These registers are always used when affinity routing is not enabled. When affinity routing is enabled for the Secure state, GICD\_NSACR0 is RES0 and [GICR\\_NSACR](#) provides equivalent functionality for SGIs.

These registers do not support PPIs, therefore GICD\_NSACR1 is RAZ/WI.

## Configuration

Some or all RW fields of this register have defined reset values.

The concept of selective enabling of Non-secure access to Group 0 and Secure Group 1 interrupts applies to SGIs and SPIs.

GICD\_NSACR0 is a Banked register used for SGIs. A copy is provided for every PE that has a CPU interface and that supports this feature.

## Attributes

GICD\_NSACR<n> is a 32-bit register.

## Field descriptions

The GICD\_NSACR<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
																NS_access<x>, bits [2x+1:2x], for x = 0 to 15															

### NS\_access<x>, bits [2x+1:2x], for x = 0 to 15

Controls Non-secure access of the interrupt with ID 16n + x.

If the corresponding interrupt does not support configurable Non-secure access, the field is RAZ/WI.

Otherwise, the field is RW and determines the level of Non-secure control permitted if the interrupt is a Secure interrupt. If the interrupt is a Non-secure interrupt, this field is ignored.

The possible values of each 2-bit field are:



NS_access<x>	Meaning
00	No Non-secure access is permitted to fields associated with the corresponding interrupt.
01	Non-secure read and write access is permitted to set-pending bits in <a href="#">GICD_ISPENDR&lt;n&gt;</a> associated with the corresponding interrupt. A Non-secure write access to <a href="#">GICD_SETSPI_NSR</a> is permitted to set the pending state of the corresponding interrupt. A Non-secure write access to <a href="#">GICD_SGIR</a> is permitted to generate a Secure SGI for the corresponding interrupt. An implementation might also provide read access to clear-pending bits in <a href="#">GICD_ICPENDR&lt;n&gt;</a> associated with the corresponding interrupt.
10	As 01, but adds Non-secure read and write access permission to fields associated with the corresponding interrupt in the <a href="#">GICD_ICPENDR&lt;n&gt;</a> registers. A Non-secure write access to <a href="#">GICD_CLRSPI_NSR</a> is permitted to clear the pending state of the corresponding interrupt. Also adds Non-secure read access permission to fields associated with the corresponding interrupt in the <a href="#">GICD_ISACTIVER&lt;n&gt;</a> and <a href="#">GICD_ICACTIVER&lt;n&gt;</a> registers.
11	For GICD_NSACR0 this encoding is reserved and treated as 10. For all other GICD_NSACR<n> registers this encoding is treated as 10, but adds Non-secure read and write access permission to <a href="#">GICD_ITARGETSR&lt;n&gt;</a> and <a href="#">GICD_IROUTER&lt;n&gt;</a> fields associated with the corresponding interrupt.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

For interrupt ID m, when DIV and MOD are the integer division and modulo operations:

- The corresponding GICD\_NSACR<n> number, n, is given by  $n = m \text{ DIV } 16$ .
- The offset of the required GICD\_NSACR<n> register is  $(0 \times E00 + (4 * n))$ .

#### Note

Because each field in this register comprises two bits, GICD\_NSACR0 controls access rights to SGI registers, GICD\_NSACR1 controls access to PPI registers (and is always RAZ/WI), and all other GICD\_NSACR<n> registers control access to SPI registers.

For compatibility with GICv2, writes to GICD\_NSACR0 for a particular PE must be coordinated within the Distributor and must update [GICR\\_NSACR](#) for the Redistributor associated with that PE.

## Accessing the GICD\_NSACR<n>

GICD\_NSACR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 0E00 + 4n$

# GICD\_SETSPI\_NSR, Set Non-secure SPI Pending Register

The GICD\_SETSPI\_NSR characteristics are:

## Purpose

Adds the pending state to a valid SPI if permitted by the Security state of the access and the [GICD\\_NSACR<n>](#) value for that SPI.

A write to this register changes the state of an inactive SPI to pending, and the state of an active SPI to active and pending.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

Writes to this register have no effect if:

- The value written specifies a Secure SPI, the value is written by a Non-secure access, and the value of the corresponding [GICD\\_NSACR<n>](#) register is 0.
- The value written specifies an invalid SPI.
- The SPI is already pending.

16-bit accesses to bits [15:0] of this register must be supported.

### Note

A Secure access to this register can set the pending state of any valid SPI.

## Configuration

If [GICD\\_TYPER](#).MBIS == 0, this register is reserved.

When [GICD\\_CTLR](#).DS==1, this register provides functionality for all SPIs.

## Attributes

GICD\_SETSPI\_NSR is a 32-bit register.

## Field descriptions

The GICD\_SETSPI\_NSR 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										

INTID

### Bits [31:10]

Reserved, RES0.

**INTID, bits [9:0]**

The INTID of the SPI.

The function of this register depends on whether the targeted SPI is configured to be an edge-triggered or level-sensitive interrupt:

- For an edge-triggered interrupt, a write to GICD\_SETSPI\_NSR or [GICD\\_SETSPI\\_SR](#) adds the pending state to the targeted interrupt. It will stop being pending on activation, or if the pending state is removed by a write to [GICD\\_CLRSPI\\_NSR](#), [GICD\\_CLRSPI\\_SR](#), or [GICD\\_ICPENDR<n>](#).
- For a level-sensitive interrupt, a write to GICD\_SETSPI\_NSR or [GICD\\_SETSPI\\_SR](#) adds the pending state to the targeted interrupt. It will remain pending until it is deasserted by a write to [GICD\\_CLRSPI\\_NSR](#) or [GICD\\_CLRSPI\\_SR](#). If the interrupt is activated between having the pending state added and being deactivated, then the interrupt will be active and pending.

## Accessing the GICD\_SETSPI\_NSR

GICD\_SETSPI\_NSR can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	0x0040

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_SETSPI\_SR, Set Secure SPI Pending Register

The GICD\_SETSPI\_SR characteristics are:

## Purpose

Adds the pending state to a valid SPI.

A write to this register changes the state of an inactive SPI to pending, and the state of an active SPI to active and pending.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WI	WO	WI

Writes to this register have no effect if:

- The value is written by a Non-secure access.
- The value written specifies an invalid SPI.
- The SPI is already pending.

16-bit accesses to bits [15:0] of this register must be supported.

## Configuration

If [GICD\\_TYPER](#).MBIS == 0, this register is reserved.

When [GICD\\_CTLR](#).DS==1, this register is WI.

## Attributes

GICD\_SETSPI\_SR is a 32-bit register.

## Field descriptions

The GICD\_SETSPI\_SR 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	INTID									

### Bits [31:10]

Reserved, RES0.

### INTID, bits [9:0]

The INTID of the SPI.

The function of this register depends on whether the targeted SPI is configured to be an edge-triggered or level-sensitive interrupt:

- For an edge-triggered interrupt, a write to [GICD\\_SETSPI\\_NSR](#) or GICD\_SETSPI\_SR adds the pending state to the targeted interrupt. It will stop being pending on activation, or if the pending state is removed by a write to [GICD\\_CLRSPI\\_NSR](#), [GICD\\_CLRSPI\\_SR](#), or [GICD\\_ICPENDR<n>](#).

- For a level-sensitive interrupt, a write to [GICD\\_SETSPI\\_NSR](#) or GICD\_SETSPI\_SR adds the pending state to the targeted interrupt. It will remain pending until it is deasserted by a write to [GICD\\_CLRSPI\\_NSR](#) or [GICD\\_CLRSPI\\_SR](#). If the interrupt is activated between having the pending state added and being deactivated, then the interrupt will be active and pending.

## Accessing the GICD\_SETSPI\_SR

GICD\_SETSPI\_SR can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	0x0050

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_SGIR, Software Generated Interrupt Register

The GICD\_SGIR characteristics are:

## Purpose

Controls the generation of SGIs.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

This register is used only when affinity routing is not enabled. When affinity routing is enabled, this register is RES0.

It is IMPLEMENTATION DEFINED whether this register has any effect when the forwarding of interrupts by the Distributor is disabled by [GICD\\_CTLR](#).

## Configuration

This register is available in all configurations of the GIC. If the GIC supports two Security states this register is Common.

## Attributes

GICD\_SGIR is a 32-bit register.

## Field descriptions

The GICD\_SGIR 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	TargetListFilter					CPU	TargetList				NSATT	0	0	0	0	0	0	0	0	0	0	0			INTID	

### Bits [31:26]

Reserved, RES0.

### TargetListFilter, bits [25:24]

Determines how the Distributor processes the requested SGI.

TargetListFilter	Meaning
00	Forward the interrupt to the CPU interfaces specified by GICD_SGIR.CPUTargetList.
01	Forward the interrupt to all CPU interfaces except that of the PE that requested the interrupt.
10	Forward the interrupt only to the CPU interface of the PE that requested the interrupt.
11	Reserved.

**CPUTargetList, bits [23:16]**

When GICD\_SGIR.TargetListFilter is 00, this field defines the CPU interfaces to which the Distributor must forward the interrupt.

Each bit of the field refers to the corresponding CPU interface. For example, CPUTargetList[0] corresponds to interface 0. Setting a bit to 1 indicates that the interrupt must be forwarded to the corresponding interface.

If this field is 00000000 when GICD\_SGIR.TargetListFilter is 00, the Distributor does not forward the interrupt to any CPU interface.

**NSATT, bit [15]**

Specifies the required group of the SGI.

NSATT	Meaning
0	Forward the SGI specified in the INTID field to a specified CPU interface only if the SGI is configured as Group 0 on that interface.
1	Forward the SGI specified in the INTID field to a specified CPU interface only if the SGI is configured as Group 1 on that interface.

This field is writable only by a Secure access. Non-secure accesses can also generate Group 0 interrupts, if allowed to do so by GICD\_NSACR0. Otherwise, Non-secure writes to GICD\_SGIR generate an SGI only if the specified SGI is programmed as Group 1, regardless of the value of bit [15] of the write.

**Bits [14:4]**

Reserved, RES0.

**INTID, bits [3:0]**

The INTID of the SGI to forward to the specified CPU interfaces.

**Accessing the GICD\_SGIR**

GICD\_SGIR can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	0x0F00

# GICD\_SPENDSGIR<n>, SGI Set-Pending Registers, n = 0 - 3

The GICD\_SPENDSGIR<n> characteristics are:

## Purpose

Adds the pending state to an SGI.

A write to this register changes the state of an inactive SGI to pending, and the state of an active SGI to active and pending.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

These registers are used only when affinity routing is not enabled. When affinity routing is enabled for the Security state of an interrupt then the bit associated with SGI in that Security state is RES0. An implementation is permitted to make the register RAZ/WI in this case.

A register bit that corresponds to an unimplemented SGI is RAZ/WI.

These registers are byte-accessible.

If the GIC implementation supports two Security states:

- A register bit that corresponds to a Group 0 interrupt is RAZ/WI to Non-secure accesses.
- Register bits corresponding to unimplemented PEs are RAZ/WI.

## Configuration

Some or all RW fields of this register have defined reset values.

Four SGI set-pending registers are implemented. Each register contains eight set-pending bits for each of four SGIs, for a total of 16 possible SGIs.

In multiprocessor implementations, each PE has a copy of these registers.

## Attributes

GICD\_SPENDSGIR<n> is a 32-bit register.

## Field descriptions

The GICD\_SPENDSGIR<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
SGI_set_pending_bits<x>, bits [8x+7:8x], for x = 0 to 3																															

### SGI\_set\_pending\_bits<x>, bits [8x+7:8x], for x = 0 to 3

Adds the pending state to SGI number 4n + x for the PE corresponding to the bit number written to.

Reads and writes have the following behavior:



SGI_set_pending_bits<x>	Meaning
0	If read, indicates that the SGI from the corresponding PE is not pending and is not active and pending. If written, has no effect.
1	If read, indicates that the SGI from the corresponding PE is pending or is active and pending. If written, adds the pending state to the SGI for the corresponding PE.

When this register has an architecturally-defined reset value, this field resets to 0.

For SGI ID m, generated by processing element C writing to the corresponding [GICD\\_SGIR](#) field, where DIV and MOD are the integer division and modulo operations:

- The corresponding GICD\_SPENDSGIR<n> number is given by  $n = m \text{ DIV } 4$ .
- The offset of the required register is  $(0 \times F20 + (4n))$ .
- The offset of the required field within the register GICD\_CPENDSGIR<n> is given by  $m \text{ MOD } 4$ .
- The required bit in the 8-bit SGI set-pending field m is bit C.

## Accessing the GICD\_SPENDSGIR<n>

GICD\_SPENDSGIR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	$0 \times 0F20 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICD\_STATUSR, Error Reporting Status Register

The GICD\_STATUSR characteristics are:

## Purpose

Provides software with a mechanism to detect:

- Accesses to reserved locations.
- Writes to read-only locations.
- Reads of write-only locations.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

In an implementation that supports two Security states, there are separate Secure and Non-secure instances of this register:

	Security disabled	Secure	Non-secure
GICD_STATUSR(S)	RW	RW	-
GICD_STATUSR(NS)	RW	-	RW

This is an optional register. If the register is not implemented, the location is RAZ/WI.

## Configuration

If the GIC implementation supports two Security states this register is Banked to provide Secure and Non-secure copies.

## Attributes

GICD\_STATUSR is a 32-bit register.

## Field descriptions

The GICD\_STATUSR 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	WROD	RWOD	WRD	RRD

### Bits [31:4]

Reserved, RES0.

### WROD, bit [3]

Write to an RO location.

WROD	Meaning
0	Normal operation.
1	A write to an RO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**RWOD, bit [2]**

Read of a WO location.

<b>RWOD</b>	<b>Meaning</b>
0	Normal operation.
1	A read of a WO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**WRD, bit [1]**

Write to a reserved location.

<b>WRD</b>	<b>Meaning</b>
0	Normal operation.
1	A write to a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**RRD, bit [0]**

Read of a reserved location.

<b>RRD</b>	<b>Meaning</b>
0	Normal operation.
1	A read of a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**Accessing the GICD\_STATUSR**

GICD\_STATUSR can be accessed through its memory-mapped interface:

<b>Component</b>	<b>Offset</b>
GIC Distributor	0x0010

# GICD\_TYPER, Interrupt Controller Type Register

The GICD\_TYPER characteristics are:

## Purpose

Provides information about what features the GIC implementation supports. It indicates:

- Whether the GIC implementation supports two Security states.
- The maximum number of INTIDs that the GIC implementation supports.
- The number of PEs that can be used as interrupt targets.

This register is part of the GIC Distributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

## Configuration

This register is available in all configurations of the GIC. When [GICD\\_CTLR.DS](#)==0, this register is Common.

## Attributes

GICD\_TYPER is a 32-bit register.

## Field descriptions

The GICD\_TYPER 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	<a href="#">No1N</a>	<a href="#">A3V</a>		<a href="#">IDbits</a>		<a href="#">DVIS</a>	<a href="#">LPIS</a>	<a href="#">MBIS</a>	0	0	0	0	0	0	<a href="#">SecurityExtn</a>	0	0	<a href="#">CPUNumber</a>	<a href="#">ITLinesNumber</a>							

### Bits [31:26]

Reserved, RES0.

### No1N, bit [25]

Indicates whether 1 of N SPI interrupts are supported.

No1N	Meaning
0	1 of N SPI interrupts are supported.
1	1 of N SPI interrupts are not supported.

### A3V, bit [24]

Affinity 3 valid. Indicates whether the Distributor supports nonzero values of Affinity level 3. Possible values are:

A3V	Meaning
0	The Distributor only supports zero values of Affinity level 3.
1	The Distributor supports nonzero values of Affinity level 3.

**IDbits, bits [23:19]**

The number of interrupt identifier bits supported, minus one.

**DVIS, bit [18]**

Indicates whether the implementation supports Direct Virtual LPI injection.

DVIS	Meaning
0	The implementation does not support Direct Virtual LPI injection.
1	The implementation supports Direct Virtual LPI injection.

For GICv3, this field is RES0.

**LPIS, bit [17]**

Indicates whether the implementation supports LPIs.

LPIS	Meaning
0	The implementation does not support LPIs.
1	The implementation supports LPIs.

**MBIS, bit [16]**

Indicates whether the implementation supports message-based interrupts by writing to Distributor registers.

MBIS	Meaning
0	The implementation does not support message-based interrupts by writing to Distributor registers. The <a href="#">GICD_CLRSPI_NSR</a> , <a href="#">GICD_SETSPI_NSR</a> , <a href="#">GICD_CLRSPI_SR</a> , and <a href="#">GICD_SETSPI_SR</a> registers are reserved.
1	The implementation supports message-based interrupts by writing to the <a href="#">GICD_CLRSPI_NSR</a> , <a href="#">GICD_SETSPI_NSR</a> , <a href="#">GICD_CLRSPI_SR</a> , or <a href="#">GICD_SETSPI_SR</a> registers.

**Bits [15:11]**

Reserved, RES0.

**SecurityExtn, bit [10]**

Indicates whether the GIC implementation supports two Security states:

When [GICD\\_CTLR.DS](#) == 1, this field is RAZ.

SecurityExtn	Meaning
0	The GIC implementation supports only a single Security state.
1	The GIC implementation supports two Security states.

**Bits [9:8]**

Reserved, RES0.

**CPUNumber, bits [7:5]**

Reports the number of PEs that can be used when affinity routing is not enabled, minus 1.

These PEs must be numbered contiguously from zero, but the relationship between this number and the affinity hierarchy from MPIDR is IMPLEMENTATION DEFINED. If the implementation does not support ARE being zero, this field is 000.

**ITLinesNumber, bits [4:0]**

Indicates the maximum SPI INTID that the GIC implementation supports. If the value of this field is N, the maximum SPI INTID is 32(N+1)-1. For example, 00011 specifies that the maximum SPI INTID is 127.

The maximum SPI INTID an implementation might support is 1019 (field value 11111). Regardless of the range of INTIDs defined by this field, interrupt IDs 1020-1023 are reserved for special purposes.

**Note**

The value derived from this field specifies the maximum number of SPIs that the GIC implementation might support. An implementation might not implement all SPIs up to this maximum.

The ITLinesNumber field only indicates the maximum number of SPIs that the GIC implementation might support. This value determines the number of instances of the following interrupt registers:

- [GICD\\_IGROUPR<n>](#).
- [GICD\\_ISENBALER<n>](#).
- [GICD\\_ICENABLER<n>](#).
- [GICD\\_ISPENDR<n>](#).
- [GICD\\_ICPENDR<n>](#).
- [GICD\\_ISACTIVER<n>](#).
- [GICD\\_ICACTIVER<n>](#).
- [GICD\\_IPRIORITYR<n>](#).
- [GICD\\_ITARGETSR<n>](#).
- [GICD\\_ICFGR<n>](#).

The GIC architecture does not require a GIC implementation to support a continuous range of SPI interrupt IDs. Software must check which SPI INTIDs are supported, up to the maximum value indicated by GICD\_TYPER.ITLinesNumber.

**Accessing the GICD\_TYPER**

GICD\_TYPER can be accessed through its memory-mapped interface:

Component	Offset
GIC Distributor	0x0004

# GICH\_APR<n>, Active Priorities Registers, n = 0 - 3

The GICH\_APR<n> characteristics are:

## Purpose

These registers track which preemption levels are active in the virtual CPU interface, and indicate the current active priority. Corresponding bits are set to 1 in this register when an interrupt is acknowledged, based on [GICH\\_LR<n>.](#)Priority, and the least significant bit set is cleared on EOI.

This register is part of the GIC virtualised guest interface control registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

These registers are used only when System register access is not enabled. When System register access is enabled the following registers provide equivalent functionality:

- In AArch64:
  - For Group 0, [ICH\\_AP0R<n>\\_EL2.](#)
  - For Group 1, [ICH\\_AP1R<n>\\_EL2.](#)
- In AArch32:
  - For Group 0, [ICH\\_AP0R<n>.](#)
  - For Group 1, [ICH\\_AP1R<n>.](#)

## Configuration

Some or all RW fields of this register have defined reset values.

This register is available when the GIC implementation supports interrupt virtualization.

The number of registers required depends on how many bits are implemented in [GICH\\_LR<n>.](#)Priority:

- When 5 priority bits are implemented, 1 register is required (GICH\_APR0).
- When 6 priority bits are implemented, 2 registers are required (GICH\_APR0, GICH\_APR1).
- When 7 priority bits are implemented, 4 registers are required (GICH\_APR0, GICH\_APR1, GICH\_APR2, GICH\_APR3).

Unimplemented registers are RAZ/WI.

## Attributes

GICH\_APR<n> is a 32-bit register.

## Field descriptions

The GICH\_APR<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
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

P<x>, bit [x], for x = 0 to 31

Active priorities. Possible values of each bit are:

P<x>	Meaning
0	There is no interrupt active at the priority corresponding to that bit.
1	There is an interrupt active at the priority corresponding to that bit.

The correspondence between priorities and bits depends on the number of bits of priority that are implemented.

If 5 bits of priority are implemented (bits [7:3] of priority), then there are 32 priority groups, and the active state of these priorities are held in GICH\_APR0 in the bits corresponding to Priority[7:3].

If 6 bits of priority are implemented (bits [7:2] of priority), then there are 64 priority groups, and:

- The active state of priorities 0 - 124 are held in GICH\_APR0 in the bits corresponding to 0:Priority[6:2].
- The active state of priorities 128 - 252 are held in GICH\_APR1 in the bits corresponding to 1:Priority[6:2].

If 7 bits of priority are implemented (bits [7:1] of priority), then there are 128 priority groups, and:

- The active state of priorities 0 - 62 are held in GICH\_APR0 in the bits corresponding to 00:Priority[5:1].
- The active state of priorities 64 - 126 are held in GICH\_APR1 in the bits corresponding to 01:Priority[5:1].
- The active state of priorities 128 - 190 are held in GICH\_APR2 in the bits corresponding to 10:Priority[5:1].
- The active state of priorities 192 - 254 are held in GICH\_APR3 in the bits corresponding to 11:Priority[5:1].

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the GICH\_APR<n>

GICH\_APR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual interface control	0x00F0 + 4n

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# GICH\_EISR, End Interrupt Status Register

The GICH\_EISR characteristics are:

## Purpose

Indicates which List registers have outstanding EOI maintenance interrupts.

This register is part of the GIC virtualised guest interface control registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICH\\_EISR](#) provides equivalent functionality.
- For AArch64 implementations, [ICH\\_EISR\\_EL2](#) provides equivalent functionality.

Bits corresponding to unimplemented List registers are RAZ.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICH\_EISR is a 32-bit register.

## Field descriptions

The GICH\_EISR 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	Status<n>, bit [n], for n = 0 to 15															

### Bits [31:16]

Reserved, RES0.

### Status<n>, bit [n], for n = 0 to 15

EOI maintenance interrupt status for List register <n>:

Status<n>	Meaning
0	<a href="#">GICH_LR&lt;n&gt;</a> does not have an EOI maintenance interrupt.
1	<a href="#">GICH_LR&lt;n&gt;</a> has an EOI maintenance interrupt that has not been handled.

For any [GICH\\_LR<n>](#) register, the corresponding status bit is set to 1 if all of the following are true:

- [GICH\\_LR<n>](#).State is 0b00.

- [GICH\\_LR<n>](#).HW == 0.
- [GICH\\_LR<n>](#).EOI == 1.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICH\_EISR

GICH\_EISR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual interface control	0x0020

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICH\_ELRSR, Empty List Register Status Register

The GICH\_ELRSR characteristics are:

## Purpose

Indicates which List registers contain valid interrupts.

This register is part of the GIC virtualised guest interface control registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICH\\_ELRSR](#) provides equivalent functionality.
- For AArch64 implementations, [ICH\\_ELRSR\\_EL2](#) provides equivalent functionality.

Bits corresponding to unimplemented List registers are RES0.

## Configuration

Some or all RW fields of this register have defined reset values.

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICH\_ELRSR is a 32-bit register.

## Field descriptions

The GICH\_ELRSR 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	Status<n>, bit [n], for n = 0 to 15															

### Bits [31:16]

Reserved, RES0.

### Status<n>, bit [n], for n = 0 to 15

Status bit for List register <n>:

Status<n>	Meaning
0	<a href="#">GICH_LR&lt;n&gt;</a> , if implemented, contains a valid interrupt. Using this List register can result in overwriting a valid interrupt.
1	<a href="#">GICH_LR&lt;n&gt;</a> does not contain a valid interrupt. The List register is empty and can be used without overwriting a valid interrupt or losing an EOI maintenance interrupt.

For any [GICH\\_LR<n>](#) register, the corresponding status bit is set to 1 if [GICH\\_LR<n>](#).State is 0b00 and either:

- [GICH\\_LR<n>](#).HW == 1.
- [GICH\\_LR<n>](#).EOI == 0.

When this register has an architecturally-defined reset value, this field resets to 1.

## Accessing the GICH\_ELRSR

GICH\_ELRSR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual interface control	0x0030

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICH\_HCR, Hypervisor Control Register

The GICH\_HCR characteristics are:

## Purpose

Controls the virtual CPU interface.

This register is part of the GIC virtualised guest interface control registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICH\\_HCR](#) provides equivalent functionality.
- For AArch64 implementations, [ICH\\_HCR\\_EL2](#) provides equivalent functionality.

GICH\_HCR.En must be set to 1 for any virtual or maintenance interrupt to be asserted.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICH\_HCR is a 32-bit register.

## Field descriptions

The GICH\_HCR 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
EOICount								0								VGrp1DIE		VGrp1EIE		VGrp0DIE		VGrp0EIE		NPIE		LRENPIE		UIE		En	

### EOICount, bits [31:27]

Counts the number of EOIs received that do not have a corresponding entry in the List registers. The virtual CPU interface increments this field automatically when a matching EOI is received. EOIs that do not clear a bit in [GICH\\_APR<n>](#) do not cause an increment. If an EOI occurs when the value of this field is 31, then the field wraps to 0.

The maintenance interrupt is asserted whenever this field is nonzero and  $GICH\_HCR.LRENPIE == 1$ .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### Bits [26:8]

Reserved, RES0.

**VGrp1DIE, bit [7]**

VM Group 1 Disabled Interrupt Enable.

Enables the signaling of a maintenance interrupt while signaling of Group 1 interrupts from the virtual CPU interface to the connected virtual machine is disabled:

VGrp1DIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled when <a href="#">GICV_CTLR.EnableGrp1</a> == 0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**VGrp1EIE, bit [6]**

VM Group 1 Enabled Interrupt Enable.

Enables the signaling of a maintenance interrupt while signaling of Group 1 interrupts from the virtual CPU interface to the connected virtual machine is enabled:

VGrp1EIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled when <a href="#">GICV_CTLR.EnableGrp1</a> == 1.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**VGrp0DIE, bit [5]**

VM Group 0 Disabled Interrupt Enable.

Enables the signaling of a maintenance interrupt while signaling of Group 0 interrupts from the virtual CPU interface to the connected virtual machine is disabled:

VGrp0DIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled when <a href="#">GICV_CTLR.EnableGrp0</a> == 0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**VGrp0EIE, bit [4]**

VM Group 0 Enabled Interrupt Enable.

Enables the signaling of a maintenance interrupt while signaling of Group 0 interrupts from the virtual CPU interface to the connected virtual machine is enabled:

VGrp0EIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled when <a href="#">GICV_CTLR.EnableGrp0</a> == 1.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**NPIE, bit [3]**

No Pending Interrupt Enable.

Enables the signaling of a maintenance interrupt while no pending interrupts are present in the List registers:

NPIE	Meaning
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled while the List registers contain no interrupts in the pending state.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**LRENPIE, bit [2]**

List Register Entry Not Present Interrupt Enable.

Enables the signaling of a maintenance interrupt while the virtual CPU interface does not have a corresponding valid List register for an EOI request:

<b>LRENPIE</b>	<b>Meaning</b>
0	Maintenance interrupt disabled.
1	Maintenance interrupt signaled while GICH_HCR.EOICount is not 0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**UIE, bit [1]**

Underflow Interrupt Enable.

Enables the signaling of a maintenance interrupt when the List registers are either empty or hold only one valid entry.

<b>UIE</b>	<b>Meaning</b>
0	Maintenance interrupt disabled.
1	A maintenance interrupt is signaled if zero or one of the List register entries are marked as a valid interrupt.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**En, bit [0]**

Enable.

Global enable bit for the virtual CPU interface.

<b>En</b>	<b>Meaning</b>
0	Virtual CPU interface operation is disabled.
1	Virtual CPU interface operation is enabled.

When this field is 0:

- The virtual CPU interface does not signal any maintenance interrupts.
- The virtual CPU interface does not signal any virtual interrupts.
- A read of [GICV\\_IAR](#) or [GICV\\_AIAR](#) returns a spurious interrupt ID.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

The VGrp1DIE, VGrp1EIE, VGrp0DIE, and VGrp0EIE fields permit the hypervisor to track the virtual CPU interfaces that are enabled. The hypervisor can then route interrupts that have multiple targets correctly and efficiently, without having to read the virtual CPU interface status.

See Maintenance interrupts and [GICH\\_MISR](#) for more information.

**Accessing the GICH\_HCR**

GICH\_HCR can be accessed through its memory-mapped interface:

<b>Component</b>	<b>Offset</b>
GIC Virtual interface control	0x0000

# GICH\_LR<n>, List Registers, n = 0 - 15

The GICH\_LR<n> characteristics are:

## Purpose

These registers provide context information for the virtual CPU interface.

This register is part of the GIC virtualised guest interface control registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICH\\_LR<n>](#) provides equivalent functionality.
- For AArch64 implementations, [ICH\\_LR<n>\\_EL2](#) provides equivalent functionality.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

This register is available when the GIC implementation supports interrupt virtualization.

A maximum of 16 List registers can be provided. [GICH\\_VTR](#).ListRegs defines the number implemented. Unimplemented List registers are RAZ/WI.

## Attributes

GICH\_LR<n> is a 32-bit register.

## Field descriptions

The GICH\_LR<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
HWGroup		State	Priority						000			pINTID										vINTID									

### HW, bit [31]

Indicates whether this virtual interrupt is a hardware interrupt, meaning that it corresponds to a physical interrupt. Deactivation of the virtual interrupt also causes the deactivation of the physical interrupt corresponding to the INTID:

HW	Meaning
0	This interrupt is triggered entirely in software. No notification is sent to the Distributor when the virtual interrupt is deactivated.
1	A hardware interrupt. A deactivate interrupt request is sent to the Distributor when the virtual interrupt is deactivated, using GICH_LR<n>.pINTID to indicate the physical interrupt identifier. If <a href="#">GICV_CTLR</a> .EOImode == 0, this request corresponds to a write to <a href="#">GICV_EOIR</a> or <a href="#">GICV_AEOIR</a> , otherwise it corresponds to a write to <a href="#">GICV_DIR</a> .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.



**Group, bit [30]**

Indicates whether the interrupt is Group 0 or Group 1:

Group	Meaning
0	Group 0 virtual interrupt. <a href="#">GICV_CTLR.FIQEn</a> determines whether it is signaled as a virtual IRQ or as a virtual FIQ, and <a href="#">GICV_CTLR.EnableGrp0</a> enables signaling of this interrupt to the virtual machine.
1	Group 1 virtual interrupt, signaled as a virtual IRQ. <a href="#">GICV_CTLR.EnableGrp1</a> enables signaling of this interrupt to the virtual machine.

**Note**

[GICV\\_CTLR.CBPR](#) controls whether [GICV\\_BPR](#) or [GICV\\_ABPR](#) determines if a pending Group 1 interrupt has sufficient priority to preempt current execution.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**State, bits [29:28]**

The state of the interrupt. This field has one of the following values:

State	Meaning
00	Inactive
01	Pending
10	Active
11	Active and pending

The GIC updates these state bits as virtual interrupts proceed through the interrupt life cycle. Entries in the inactive state are ignored, except for the purpose of generating virtual maintenance interrupts.

**Note**

For hardware interrupts, the active and pending state is held in the Distributor rather than the virtual CPU interface. A hypervisor must only use the active and pending state for software originated interrupts, which are typically associated with virtual devices, or for SGIs.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Priority, bits [27:23]**

The priority of this interrupt.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Bits [22:20]**

Reserved, RES0.

**pINTID, bits [19:10]**

The function of this field depends on the value of GICH\_LR<n>.HW.

When GICH\_LR<n>.HW == 0:

- Bit [19] indicates whether the interrupt triggers an EOI maintenance interrupt. If this bit is 1, then when the interrupt identified by vINTID is deactivated, an EOI maintenance interrupt is asserted.
- Bits [18:13] are reserved, SBZ.
- If the vINTID field value corresponds to an SGI (that is, 0-15), bits [12:10] contain the number of the requesting PE. This appears in the corresponding field of [GICV\\_IAR](#) or [GICV\\_AIAR](#). If the vINTID field value is not 0-15, this field must be cleared to 0.

When GICH\_LR<n>.HW == 1:

- This field indicates the pINTID that the hypervisor forwards to the Distributor. This field is only required to implement enough bits to hold a valid value for the ID configuration. Any unused higher order bits are RAZ/WI.
- If the value of pINTID is 0-15 or 1020-1023, behavior is UNPREDICTABLE. If the value of pINTID is 16-31, this field applies to the PPI associated with this same PE as the virtual CPU interface requesting the deactivation.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### vINTID, bits [9:0]

This INTID is returned to the VM when the interrupt is acknowledged through [GICV\\_IAR](#). Each valid interrupt stored in the List registers must have a unique vINTID for that virtual CPU interface. If the value of vINTID is 1020-1023, behavior is UNPREDICTABLE.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICH\_LR<n>

GICH\_LR<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual interface control	$0 \times 0100 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICH\_MISR, Maintenance Interrupt Status Register

The GICH\_MISR characteristics are:

## Purpose

Indicates which maintenance interrupts are asserted.

This register is part of the GIC virtualised guest interface control registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICH\\_MISR](#) provides equivalent functionality.
- For AArch64 implementations, [ICH\\_MISR\\_EL2](#) provides equivalent functionality.

A maintenance interrupt is asserted only if at least one bit is set to 1 in this register and if [GICH\\_HCR](#).En == 1.

## Configuration

Some or all RW fields of this register have defined reset values.

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICH\_MISR is a 32-bit register.

## Field descriptions

The GICH\_MISR 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	VGrp1D	VGrp1E	VGrp0D	VGrp0E	NPLREN	P	U	EOI

### Bits [31:8]

Reserved, RES0.

### VGrp1D, bit [7]

vPE Group 1 Disabled.

VGrp1D	Meaning
0	vPE Group 1 Disabled maintenance interrupt not asserted.
1	vPE Group 1 Disabled maintenance interrupt asserted.

This maintenance interrupt is asserted when [GICH\\_HCR](#).VGrp1DIE == 1 and [GICH\\_VMCR](#).VENG1 == 0.

When this register has an architecturally-defined reset value, this field resets to 0.

**VGrp1E, bit [6]**

vPE Group 1 Enabled.

VGrp1E	Meaning
0	vPE Group 1 Enabled maintenance interrupt not asserted.
1	vPE Group 1 Enabled maintenance interrupt asserted.

This maintenance interrupt is asserted when [GICH\\_HCR.VGrp1EIE](#) == 1 and [GICH\\_VMCR.VENG1](#) == 1.

When this register has an architecturally-defined reset value, this field resets to 0.

**VGrp0D, bit [5]**

vPE Group 0 Disabled.

VGrp0D	Meaning
0	vPE Group 0 Disabled maintenance interrupt not asserted.
1	vPE Group 0 Disabled maintenance interrupt asserted.

This maintenance interrupt is asserted when [GICH\\_HCR.VGrp0DIE](#) == 1 and [GICH\\_VMCR.VENG0](#) == 0.

When this register has an architecturally-defined reset value, this field resets to 0.

**VGrp0E, bit [4]**

vPE Group 0 Enabled.

VGrp0E	Meaning
0	vPE Group 0 Enabled maintenance interrupt not asserted.
1	vPE Group 0 Enabled maintenance interrupt asserted.

This maintenance interrupt is asserted when [GICH\\_HCR.VGrp0EIE](#) == 1 and [GICH\\_VMCR.VENG0](#) == 1.

When this register has an architecturally-defined reset value, this field resets to 0.

**NP, bit [3]**

No Pending.

NP	Meaning
0	No Pending maintenance interrupt not asserted.
1	No Pending maintenance interrupt asserted.

This maintenance interrupt is asserted when [GICH\\_HCR.NPIE](#) == 1 and no List register is in the pending state.

When this register has an architecturally-defined reset value, this field resets to 0.

**LREN, bit [2]**

List Register Entry Not Present.

LREN	Meaning
0	List Register Entry Not Present maintenance interrupt not asserted.
1	List Register Entry Not Present maintenance interrupt asserted.

This maintenance interrupt is asserted when [GICH\\_HCR.LRENPIE](#) == 1 and [GICH\\_HCR.EOICount](#) is nonzero.

When this register has an architecturally-defined reset value, this field resets to 0.

**U, bit [1]**

Underflow.

U	Meaning
0	Underflow maintenance interrupt not asserted.
1	Underflow maintenance interrupt asserted.

This maintenance interrupt is asserted when [GICH\\_HCR](#).UIE == 1 and zero or one of the List register entries are marked as a valid interrupt.

When this register has an architecturally-defined reset value, this field resets to 0.

## EOI, bit [0]

End Of Interrupt.

EOI	Meaning
0	End Of Interrupt maintenance interrupt not asserted.
1	End Of Interrupt maintenance interrupt asserted.

This maintenance interrupt is asserted when at least one bit in [GICH\\_EISR](#) == 1.

When this register has an architecturally-defined reset value, this field resets to 0.

---

### Note

A List register is in the pending state only if the corresponding [GICH\\_LR<n>](#) value is 01, that is, pending. The active and pending state is not included.

---

## Accessing the GICH\_MISR

GICH\_MISR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual interface control	0x0010

# GICH\_VMCR, Virtual Machine Control Register

The GICH\_VMCR characteristics are:

## Purpose

Enables the hypervisor to save and restore the virtual machine view of the GIC state. This register is updated when a virtual machine updates the virtual CPU interface registers.

This register is part of the GIC virtualised guest interface control registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICH\\_VMCR](#) provides equivalent functionality.
- For AArch64 implementations, [ICH\\_VMCR\\_EL2](#) provides equivalent functionality.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICH\_VMCR is a 32-bit register.

## Field descriptions

The GICH\_VMCR 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																												
VPMR								VBPR0		VBPR1		0				0				0				0				VEOIM				0				0				VCBPR				VFIQEn				VAcKCtl				VENG1				VENG0			

### VPMR, bits [31:24]

Virtual priority mask. The priority mask level for the CPU interface. If the priority of an interrupt is higher than the value indicated by this field, the interface signals the interrupt to the PE.

This alias field is updated when a VM updates [GICV\\_PMR](#).Priority.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### VBPR0, bits [23:21]

Virtual Binary Point Register, Group 0. Defines the point at which the priority value fields split into two parts, the Group priority field and the subpriority field. The Group priority field determines Group 0 interrupt preemption, and also determines Group 1 interrupt preemption if  $GICH\_VMCR.VCBPR == 1$ .

This alias field is updated when a VM updates [GICV\\_BPR](#).Binary\_Point.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### VBPR1, bits [20:18]

Virtual Binary Point Register, Group 1. Defines the point at which the priority value fields split into two parts, the Group priority field and the subpriority field. The Group priority field determines Group 1 interrupt preemption if `GICH_VMCR.VCBPR == 0`.

This alias field is updated when a VM updates [GICV\\_ABPR.Binary\\_Point](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### Bits [17:10]

Reserved, RES0.

### VEOIM, bit [9]

Virtual EOImode. Possible values of this bit are:

VEOIM	Meaning
0	A write of an INTID to <a href="#">GICV_EOIR</a> or <a href="#">GICV_AEOIR</a> drops the priority of the interrupt with that INTID, and also deactivates that interrupt.
1	A write of an INTID to <a href="#">GICV_EOIR</a> or <a href="#">GICV_AEOIR</a> only drops the priority of the interrupt with that INTID. Software must write to <a href="#">GICV_DIR</a> to deactivate the interrupt.

This alias field is updated when a VM updates [GICV\\_CTLR.EOImode](#).

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 [8:5]

Reserved, RES0.

### VCBPR, bit [4]

Virtual Common Binary Point Register. Possible values of this bit are:

VCBPR	Meaning
0	<a href="#">GICV_ABPR</a> determines the preemption group for Group 1 interrupts.
1	<a href="#">GICV_BPR</a> determines the preemption group for Group 1 interrupts.

This alias field is updated when a VM updates [GICV\\_CTLR.CBPR](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### VFIQEn, bit [3]

Virtual FIQ enable. Possible values of this bit are:

VFIQEn	Meaning
0	Group 0 virtual interrupts are presented as virtual IRQs.
1	Group 0 virtual interrupts are presented as virtual FIQs.

This alias field is updated when a VM updates [GICV\\_CTLR.FIQEn](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### VAckCtl, bit [2]

Virtual AckCtl. Possible values of this bit are:

VAckCtl	Meaning
0	If the highest priority pending interrupt is Group 1, a read of <a href="#">GICV_IAR</a> or <a href="#">GICV_HPPIR</a> returns an INTID of 1022.
1	If the highest priority pending interrupt is Group 1, a read of <a href="#">GICV_IAR</a> or <a href="#">GICV_HPPIR</a> returns the INTID of the corresponding interrupt.

This alias field is updated when a VM updates [GICV\\_CTLR.AckCtl](#).

This field is supported for backwards compatibility with GICv2. ARM deprecates the use of this field.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### VENG1, bit [1]

Virtual interrupt enable, Group 1. Possible values of this bit are:

VENG1	Meaning
0	Group 1 virtual interrupts are disabled.
1	Group 1 virtual interrupts are enabled.

This alias field is updated when a VM updates [GICV\\_CTLR.EnableGrp1](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### VENG0, bit [0]

Virtual interrupt enable, Group 0. Possible values of this bit are:

VENG0	Meaning
0	Group 0 virtual interrupts are disabled.
1	Group 0 virtual interrupts are enabled.

This alias field is updated when a VM updates [GICV\\_CTLR.EnableGrp0](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Note

A List register is in the pending state only if the corresponding [GICH\\_LR<n>](#) value is 01, that is, pending. The active and pending state is not included.

## Accessing the GICH\_VMCR

GICH\_VMCR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual interface control	0x0008



# GICH\_VTR, Virtual Type Register

The GICH\_VTR characteristics are:

## Purpose

Indicates the number of implemented virtual priority bits and List registers.

This register is part of the GIC virtualised guest interface control registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICH\\_VTR](#) provides equivalent functionality.
- For AArch64 implementations, [ICH\\_VTR\\_EL2](#) provides equivalent functionality.

## Configuration

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICH\_VTR is a 32-bit register.

## Field descriptions

The GICH\_VTR 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
<a href="#">PRIBits</a>	<a href="#">PREbits</a>	<a href="#">IDbits</a>	<a href="#">SEISA3V</a>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	<a href="#">ListRegs</a>					

### PRIBits, bits [31:29]

The number of virtual priority bits implemented, minus one.

An implementation must implement at least 32 levels of virtual priority (5 priority bits).

### PREbits, bits [28:26]

The number of virtual preemption bits implemented, minus one.

An implementation must implement at least 32 levels of virtual preemption priority (5 preemption bits).

The value of this field must be less than or equal to the value of GICH\_VTR.PRIBits.

### IDbits, bits [25:23]

The number of virtual interrupt identifier bits supported:

IDbits	Meaning
000	16 bits.
001	24 bits.

All other values are reserved.

### SEIS, bit [22]

SEI support. Indicates whether the virtual CPU interface supports generation of SEIs:

SEIS	Meaning
0	The virtual CPU interface logic does not support generation of SEIs.
1	The virtual CPU interface logic supports generation of SEIs.

### A3V, bit [21]

Affinity 3 valid. Possible values are:

A3V	Meaning
0	The virtual CPU interface logic only supports zero values of the Aff3 field in <a href="#">ICC_SGI0R_EL1</a> , <a href="#">ICC_SGI1R_EL1</a> , and <a href="#">ICC_ASGI1R_EL1</a> .
1	The virtual CPU interface logic supports nonzero values of the Aff3 field in <a href="#">ICC_SGI0R_EL1</a> , <a href="#">ICC_SGI1R_EL1</a> , and <a href="#">ICC_ASGI1R_EL1</a> .

### Bits [20:5]

Reserved, RES0.

### ListRegs, bits [4:0]

The number of implemented List registers, minus one.

## Accessing the GICH\_VTR

GICH\_VTR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual interface control	0x0004

# GICR\_CLRLPIR, Clear LPI Pending Register

The GICR\_CLRLPIR characteristics are:

## Purpose

Clears the pending state of the specified LPI.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

When written with a 32-bit write the data is zero-extended to 64 bits.

This register is mandatory in an implementation that supports LPIs and does not include an ITS. The functionality of this register is IMPLEMENTATION DEFINED in an implementation that does include an ITS.

Writes to this register have no effect if any of the following apply:

- [GICR\\_CTLR](#).EnableLPIs == 0.
- The pINTID value specifies an unimplemented LPI.
- The pINTID value specifies an LPI that is not pending.

## Configuration

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_CLRLPIR is a 64-bit register.

## Field descriptions

The GICR\_CLRLPIR 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
																pINTID															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:32]

Reserved, RES0.

### pINTID, bits [31:0]

The INTID of the physical LPI.

---

**Note**

---

---

The size of this field is IMPLEMENTATION DEFINED, and is specified by the [GICD\\_TYPER](#).Idbits field. Unimplemented bits are RES0.

---

## Accessing the GICR\_CLRLPIR

GICR\_CLRLPIR can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	RD_base	0x0048 - 0x004C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_CTLR, Redistributor Control Register

The GICR\_CTLR characteristics are:

## Purpose

Controls the operation of a Redistributor, and enables the signaling of LPIs by the Redistributor to the connected PE.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

## Configuration

Some or all RW fields of this register have defined reset values.

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_CTLR is a 32-bit register.

## Field descriptions

The GICR\_CTLR 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
UWP	0	0	0	0	DPG1S	DPG1NS	DPG0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RWP	0	0	Enable_LPIs

### UWP, bit [31]

Upstream Write Pending. Read-only. Indicates whether all upstream writes have been communicated to the Distributor.

UWP	Meaning
0	The effects of all upstream writes have been communicated to the Distributor, including any Generate SGI packets.
1	Not all the effects of upstream writes, including any Generate SGI packets, have been communicated to the Distributor.

### Bits [30:27]

Reserved, RES0.

### DPG1S, bit [26]

Disable Processor selection for Group 1 Secure interrupts. When [GICR\\_TYPER](#).DPGS == 1:

DPG1S	Meaning
0	A Group 1 Secure SPI configured to use the 1 of N distribution model can select this PE, if the PE is not asleep and if Secure Group 1 interrupts are enabled.
1	A Group 1 Secure SPI configured to use the 1 of N distribution model cannot select this PE.

When [GICR\\_TYPER](#).DPGS == 0 this bit is RAZ/WI.

When [GICD\\_CTLR](#).DS==1, this field is RAZ/WI. In GIC implementations that support two Security states, this field is only accessible by Secure accesses, and is RAZ/WI to Non-secure accesses.

It is IMPLEMENTATION DEFINED whether this bits affect the selection of PEs for interrupts using the 1 of N distribution model when [GICD\\_CTLR](#).ARE\_S==0.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### DPG1NS, bit [25]

Disable Processor selection for Group 1 Non-secure interrupts. When [GICR\\_TYPER](#).DPGS == 1:

DPG1NS	Meaning
0	A Group 1 Non-secure SPI configured to use the 1 of N distribution model can select this PE, if the PE is not asleep and if Non-secure Group 1 interrupts are enabled.
1	A Group 1 Non-secure SPI configured to use the 1 of N distribution model cannot select this PE.

When [GICR\\_TYPER](#).DPGS == 0 this bit is RAZ/WI.

It is IMPLEMENTATION DEFINED whether this bits affect the selection of PEs for interrupts using the 1 of N distribution model when [GICD\\_CTLR](#).ARE\_NS==0.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### DPG0, bit [24]

Disable Processor selection for Group 0 interrupts. When [GICR\\_TYPER](#).DPGS == 1:

DPG0	Meaning
0	A Group 0 SPI configured to use the 1 of N distribution model can select this PE, if the PE is not asleep and if Group 0 interrupts are enabled.
1	A Group 0 SPI configured to use the 1 of N distribution model cannot select this PE.

When [GICR\\_TYPER](#).DPGS == 0 this bit is RAZ/WI.

When [GICD\\_CTLR](#).DS==1, this field is always accessible. In GIC implementations that support two Security states, this field is RAZ/WI to Non-secure accesses.

It is IMPLEMENTATION DEFINED whether this bits affect the selection of PEs for interrupts using the 1 of N distribution model when [GICD\\_CTLR](#).ARE\_S==0.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

#### Bits [23:4]

Reserved, RES0.

#### RWP, bit [3]

Register Write Pending. This bit indicates whether a register write for the current Security state is in progress or not.

RWP	Meaning
0	The effect of all previous writes to the following registers are visible to all agents in the system: <ul style="list-style-type: none"> <li><a href="#">GICR_ICENABLER0</a></li> <li><a href="#">GICR_CTLR.DPG1S</a></li> <li><a href="#">GICR_CTLR.DPG1NS</a></li> <li><a href="#">GICR_CTLR.DPG0</a></li> </ul>
1	The effect of all previous writes to the following registers are not guaranteed by the architecture to be visible yet to the all agents in the system as the changes are still being propagated: <ul style="list-style-type: none"> <li><a href="#">GICR_ICENABLER0</a></li> <li><a href="#">GICR_CTLR.DPG1S</a></li> <li><a href="#">GICR_CTLR.DPG1NS</a></li> <li><a href="#">GICR_CTLR.DPG0</a></li> </ul>

**Bits [2:1]**

Reserved, RES0.

**Enable\_LPIs, bit [0]**

In implementations where affinity routing is enabled for the Security state:

Enable_LPIs	Meaning
0	LPI support is disabled. Any doorbell interrupt generated as a result of a write to a virtual LPI register must be discarded, and any ITS translation requests or commands involving LPIs in this Redistributor are ignored.
1	LPI support is enabled.

**Note**

If [GICR\\_TYPER.LPIS](#) == 0, this field is RES0.

If [GICD\\_CTLR.ARE\\_NS](#) is written from 1 to 0 when this bit is 1, behavior is an IMPLEMENTATION DEFINED choice between clearing [GICR\\_CTLR.Enable\\_LPIs](#) to 0 or maintaining its current value.

When affinity routing is not enabled for the Non-secure state, this bit is RES0. When a write changes this bit from 0 to 1, this bit becomes RES1 and the Redistributor must load the LPI Pending table from memory to check for any pending interrupts.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

The participation of a PE in the 1 of N distribution model for a given interrupt group is governed by the concatenation of [GICR\\_WAKER.ProcessorSleep](#), the appropriate [GICR\\_CTLR.DPG{1, 0}](#) bit, and the PE interrupt group enable. The behavior options are:

PS	DPG{1S, 1NS, 0}	Enable	PE behavior
0	0	0	The PE cannot be selected.
0	0	1	The PE can be selected.
0	1	*	The PE cannot be selected.
1	*	*	The PE cannot be selected when <a href="#">GICD_CTLR.E1NWF</a> == 0. When <a href="#">GICD_CTLR.E1NWF</a> == 1, the mechanism by which PEs are selected is IMPLEMENTATION DEFINED.

If an SPI using the 1 of N distribution model has been forwarded to the PE and a write to [GICR\\_CTLR](#) occurs that changes the DPG bit for the interrupt group of the SPI, the IRI must attempt to select a different target PE for the SPI. This might have no effect on the forwarded SPI if it has already been activated.

**Accessing the GICR\_CTLR**

[GICR\\_CTLR](#) can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	RD_base	0x0000





# GICR\_ICACTIVER0, Interrupt Clear-Active Register 0

The GICR\_ICACTIVER0 characteristics are:

## Purpose

Deactivates the corresponding SGI or PPI. These registers are used when saving and restoring GIC state.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

When affinity routing is not enabled for the Security state of an interrupt in GICR\_ICACTIVER0, the corresponding bit is RAZ/WI and equivalent functionality is provided by [GICD\\_ICACTIVER<n>](#) with n=0.

This register only applies to SGIs (bits [15:0]) and PPIs (bits [31:16]). For SPIs, this functionality is provided by [GICD\\_ICACTIVER<n>](#).

When [GICD\\_CTLR](#).DS == 0, bits corresponding to Secure SGIs and PPIs are RAZ/WI to Non-secure accesses.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_ICACTIVER0 is a 32-bit register.

## Field descriptions

The GICR\_ICACTIVER0 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
Clear_active_bit<x>, bit [x], for x = 0 to 31																															

### Clear\_active\_bit<x>, bit [x], for x = 0 to 31

Removes the active state from interrupt number x. Reads and writes have the following behavior:

Clear_active_bit<x>	Meaning
0	If read, indicates that the corresponding interrupt is not active, and is not active and pending. If written, has no effect.
1	If read, indicates that the corresponding interrupt is active, or is active and pending. If written, deactivates the corresponding interrupt, if the interrupt is active. If the interrupt is already deactivated, the write has no effect.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICR\_ICACTIVER0

GICR\_ICACTIVER0 can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	SGI_base	0x0380

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_ICENABLER0, Interrupt Clear-Enable Register 0

The GICR\_ICENABLER0 characteristics are:

## Purpose

Disables forwarding of the corresponding SGI or PPI to the CPU interfaces.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

When affinity routing is not enabled for the Security state of an interrupt in GICR\_ICENABLER0, the corresponding bit is RAZ/WI and equivalent functionality is provided by [GICD\\_ICENABLER<n>](#) with n=0.

This register only applies to SGIs (bits [15:0]) and PPIs (bits [31:16]). For SPIs, this functionality is provided by [GICD\\_ICENABLER<n>](#).

When [GICD\\_CTLR.DS](#) == 0, bits corresponding to Secure SGIs and PPIs are RAZ/WI to Non-secure accesses.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_ICENABLER0 is a 32-bit register.

## Field descriptions

The GICR\_ICENABLER0 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
<a href="#">Clear_enable_bit&lt;x&gt;, bit [x], for x = 0 to 31</a>																															

### Clear\_enable\_bit<x>, bit [x], for x = 0 to 31

For PPIs and SGIs, controls the forwarding of interrupt number x to the CPU interfaces. Reads and writes have the following behavior:

Clear_enable_bit<x>	Meaning
0	If read, indicates that forwarding of the corresponding interrupt is disabled. If written, has no effect.
1	If read, indicates that forwarding of the corresponding interrupt is enabled. If written, disables forwarding of the corresponding interrupt. After a write of 1 to this bit, a subsequent read of this bit returns 0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICR\_ICENABLER0

GICR\_ICENABLER0 can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	SGI_base	0x0180

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_ICFGR0, Interrupt Configuration Register 0

The GICR\_ICFGR0 characteristics are:

## Purpose

Determines whether the corresponding SGI is edge-triggered or level-sensitive.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This register is used when affinity routing is enabled.

When affinity routing is disabled for the Security state of an interrupt, the field for that interrupt is RES0 and an implementation is permitted to make the field RAZ/WI in this case. Equivalent functionality is provided by GICD\_ICFGR<n> with n=0.

When [GICD\\_CTLR.DS](#)=0, a register bit that corresponds to a Group 0 or Secure Group 1 interrupt is RAZ/WI to Non-secure accesses.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_ICFGR0 is a 32-bit register.

## Field descriptions

The GICR\_ICFGR0 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
Int_config<x>, bits [2x+1:2x], for x = 0 to 15																															

### Int\_config<x>, bits [2x+1:2x], for x = 0 to 15

Indicates whether the interrupt with ID 16n + x is level-sensitive or edge-triggered.

Int\_config[0] (bit [2x]) is RES0.

Possible values of Int\_config[1] (bit [2x+1]) are:

Int_config<x>	Meaning
0	Corresponding interrupt is level-sensitive.
1	Corresponding interrupt is edge-triggered.

For SGIs, Int\_config[1] is RAO/WI.

A read of this bit always returns the correct value to indicate the interrupt triggering method.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICR\_ICFGR0

GICR\_ICFGR0 can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	SGL_base	0x0C00

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_ICFGR1, Interrupt Configuration Register 1

The GICR\_ICFGR1 characteristics are:

## Purpose

Determines whether the corresponding PPI is edge-triggered or level-sensitive.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This register is used when affinity routing is enabled.

When affinity routing is disabled for the Security state of an interrupt, the field for that interrupt is RES0 and an implementation is permitted to make the field RAZ/WI in this case. Equivalent functionality is provided by GICD\_ICFGR<n> with n=1 .

For each supported PPI, it is IMPLEMENTATION DEFINED whether software can program the corresponding Int\_config field.

Software must disable an interrupt before the value of the corresponding programmable Int\_config field is changed. GIC behavior is otherwise UNPREDICTABLE.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_ICFGR1 is a 32-bit register.

## Field descriptions

The GICR\_ICFGR1 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
Int_config<x>, bits [2x+1:2x], for x = 0 to 15																															

### Int\_config<x>, bits [2x+1:2x], for x = 0 to 15

Indicates whether the interrupt with ID 16n + x is level-sensitive or edge-triggered.

Int\_config[0] (bit [2x]) is RES0.

Possible values of Int\_config[1] (bit [2x+1]) are:

Int_config<x>	Meaning
0	Corresponding interrupt is level-sensitive.
1	Corresponding interrupt is edge-triggered.

A read of this bit always returns the correct value to indicate the interrupt triggering method.

For PPIs, Int\_config[1] is programmable unless the implementation supports two Security states and the bit corresponds to a Group 0 or Secure Group 1 interrupt, in which case the bit is RAZ/WI to Non-secure accesses.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICR\_ICFGR1

GICR\_ICFGR1 can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	SGI_base	0x0C04

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# GICR\_ICPENDR0, Interrupt Clear-Pending Register 0

The GICR\_ICPENDR0 characteristics are:

## Purpose

Removes the pending state from the corresponding SGI or PPI.  
This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

When affinity routing is not enabled for the Security state of an interrupt in GICR\_ICPENDR0, the corresponding bit is RAZ/WI and equivalent functionality is provided by [GICD\\_ICPENDR<n>](#) with n=0.  
This register only applies to SGIs (bits [15:0]) and PPIs (bits [31:16]). For SPIs, this functionality is provided by [GICD\\_ICENABLER<n>](#).  
When [GICD\\_CTLR.DS](#) == 0, bits corresponding to Secure SGIs and PPIs are RAZ/WI to Non-secure accesses.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.  
A copy of this register is provided for each Redistributor.

## Attributes

GICR\_ICPENDR0 is a 32-bit register.

## Field descriptions

The GICR\_ICPENDR0 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
Clear_pending_bit<x>, bit [x], for x = 0 to 31																															

### Clear\_pending\_bit<x>, bit [x], for x = 0 to 31

Removes the pending state from interrupt number x. Reads and writes have the following behavior:

Clear_pending_bit<x>	Meaning
0	If read, indicates that the corresponding interrupt is not pending. If written, has no effect.
1	If read, indicates that the corresponding interrupt is pending, or active and pending. If written, changes the state of the corresponding interrupt from pending to inactive, or from active and pending to active. This has no effect in the following cases: <ul style="list-style-type: none"> <li>• If the interrupt is not pending and is not active and pending.</li> <li>• If the interrupt is a level-sensitive interrupt that is pending or active and pending for a reason other than a write to <a href="#">GICD_ISPENDR&lt;n&gt;</a>. In this case, if the interrupt signal continues to be asserted, the interrupt remains pending or active and pending.</li> </ul>

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICR\_ICPENDR0

GICR\_ICPENDR0 can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	SGI_base	0x0280

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_IGROUPR0, Interrupt Group Register 0

The GICR\_IGROUPR0 characteristics are:

## Purpose

Controls whether the corresponding SGI or PPI is in Group 0 or Group 1.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

When affinity routing is not enabled for the Security state of an interrupt in GICR\_IGROUPR0, the corresponding bit is RES0 and equivalent functionality is provided by [GICD\\_IGROUPR<n>](#) with n=0.

When [GICD\\_CTLR](#).DS == 0, the register is RAZ/WI to Non-secure accesses.

Bits corresponding to unimplemented interrupts are RAZ/WI.

### Note

Implementations must ensure that an interrupt that is pending at the time of the write uses either the old value or the new value and must ensure that the interrupt is neither lost nor handled more than once. The effect of the change must be visible in finite time.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

This register is available in all GIC configurations. If the GIC implementation supports two Security states, this register is Secure.

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_IGROUPR0 is a 32-bit register.

## Field descriptions

The GICR\_IGROUPR0 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
<a href="#">Redistributor_group_status_bit&lt;x&gt;, bit [x], for x = 0 to 31</a>																															

### Redistributor\_group\_status\_bit<x>, bit [x], for x = 0 to 31

Group status bit. In this register:

- Bits [31:16] are group status bits for PPIs.
- Bits [15:0] are group status bits for SGIs.

Redistributor_group_status_bit<x>	Meaning
0	When <a href="#">GICD_CTLR.DS</a> =1, the corresponding interrupt is Group 0. When <a href="#">GICD_CTLR.DS</a> =0, the corresponding interrupt is Secure.
1	When <a href="#">GICD_CTLR.DS</a> =1, the corresponding interrupt is Group 1. When <a href="#">GICD_CTLR.DS</a> =0, the corresponding interrupt is Non-secure Group 1.

When [GICD\\_CTLR.DS](#) == 0, the bit that corresponds to the interrupt is concatenated with the equivalent bit in [GICR\\_IGRPMODR0](#) to form a 2-bit field that defines an interrupt group. The encoding of this field is at [GICR\\_IGRPMODR0](#).

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

The considerations for the reset value of this register are the same as those for [GICD\\_IGROUPR<n>](#) with n=0.

## Accessing the GICR\_IGROUPR0

GICR\_IGROUPR0 can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	SGL_base	0x0080

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_IGRPMODR0, Interrupt Group Modifier Register 0

The GICR\_IGRPMODR0 characteristics are:

## Purpose

When [GICD\\_CTLR.DS](#)==0, this register together with the [GICD\\_IGROUPR<n>](#) registers, controls whether the corresponding interrupt is in:

- Secure Group 0.
- Non-secure Group 1.
- When System register access is enabled, Secure Group 1.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RES0	RW	RW

When affinity routing is not enabled for the Security state of an interrupt in GICR\_IGRPMODR0, the corresponding bit is RES0 and equivalent functionality is provided by [GICD\\_IGRPMODR<n>](#) with n=0.

This register only applies to SGIs (bits [15:0]) and PPIs (bits [31:16]). For SPIs, this functionality is provided by [GICD\\_IGRPMODR<n>](#).

When [GICD\\_CTLR.ARE\\_S](#) == 0 or [GICD\\_CTLR.DS](#) == 1, GICR\_IGRPMODR0 is RES0. An implementation can make this register RAZ/WI in this case.

When [GICD\\_CTLR.DS](#)==0, the register is RAZ/WI to Non-secure accesses.

---

### Note

Implementations must ensure that an interrupt that is pending at the time of the write uses either the old value or the new value and must ensure that the interrupt is neither lost nor handled more than once. The effect of the change must be visible in finite time.

---

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

When [GICD\\_CTLR.DS](#)==0, this register is Secure.

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_IGRPMODR0 is a 32-bit register.

## Field descriptions

The GICR\_IGRPMODR0 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
<a href="#">Group_modifier_bit&lt;x&gt;, bit [x], for x = 0 to 31</a>																															

**Group\_modifier\_bit<x>, bit [x], for x = 0 to 31**

Group modifier bit. In implementations where affinity routing is enabled for the Security state of an interrupt, the bit that corresponds to the interrupt is concatenated with the equivalent bit in [GICR\\_IGROUPR0](#) to form a 2-bit field that defines an interrupt group:

Group modifier bit	Group status bit	Definition	Short name
0	0	Secure Group 0	G0S
0	1	Non-secure Group 1	G1NS
1	0	Secure Group 1	G1S
1	1	Reserved, treated as Non-secure Group 1	-

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Accessing the GICR\_IGRPMODR0**

GICR\_IGRPMODR0 can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	SGI_base	0x0D00

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_IIDR, Redistributor Implementer Identification Register

The GICR\_IIDR characteristics are:

## Purpose

Provides information about the implementer and revision of the Redistributor.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

## Configuration

This register is available in all configurations of the GIC. If the GIC implementation supports two Security states, this register is Common.

## Attributes

GICR\_IIDR is a 32-bit register.

## Field descriptions

The GICR\_IIDR 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
ProductID								0	0	0	0	Variant			Revision			Implementer													

### ProductID, bits [31:24]

An IMPLEMENTATION DEFINED product identifier.

### Bits [23:20]

Reserved, RES0.

### Variant, bits [19:16]

An IMPLEMENTATION DEFINED variant number. Typically, this field is used to distinguish product variants, or major revisions of a product.

### Revision, bits [15:12]

An IMPLEMENTATION DEFINED revision number. Typically, this field is used to distinguish minor revisions of a product.

### Implementer, bits [11:0]

Contains the JEP106 code of the company that implemented the Redistributor:

- Bits [11:8] are the JEP106 continuation code of the implementer. For an ARM implementation, this field is 0x4.

- Bit [7] is always 0.
- Bits [6:0] are the JEP106 identity code of the implementer. For an ARM implementation, bits [7:0] are therefore 0x3B.

## Accessing the GICR\_IIDR

GICR\_IIDR can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	RD_base	0x0004

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# GICR\_INVALLR, Redistributor Invalidate All Register

The GICR\_INVALLR characteristics are:

## Purpose

Invalidates any cached configuration data of all physical LPIs, causing the GIC to reload the interrupt configuration from the physical LPI Configuration table at the address specified by [GICR\\_PROPBASER](#).

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

This register is mandatory in an implementation that supports LPIs and does not include an ITS. The functionality is IMPLEMENTATION DEFINED in an implementation that does include an ITS.

Writes to this register have no effect if no physical LPIs are currently stored in the local Redistributor cache.

## Configuration

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_INVALLR is a 64-bit register.

## Field descriptions

The GICR\_INVALLR 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
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 [63:0]

Reserved, RES0.

### Note

If any LPI has been forwarded to the PE and a valid write to GICR\_INVALLR is received, the Redistributor must ensure it reloads its properties from memory. This has no effect on the forwarded LPI if it has already been activated.

## Accessing the GICR\_INVALLR

GICR\_INVALLR can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	RD_base	0x00B0 - 0x00B4

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_INVLPIR, Redistributor Invalidate LPI Register

The GICR\_INVLPIR characteristics are:

## Purpose

Invalidates the cached configuration data of a specified LPI, causing the GIC to reload the interrupt configuration from the physical LPI Configuration table at the address specified by [GICR\\_PROPBASER](#).

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

When written with a 32-bit write the data is zero-extended to 64 bits.

This register is mandatory in an implementation that supports LPIs and does not include an ITS. The functionality is IMPLEMENTATION DEFINED in an implementation that does include an ITS.

Writes to this register have no effect if either:

- The specified LPI is not currently stored in the local Redistributor.
- The pINTID field corresponds to an unimplemented LPI.

## Configuration

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_INVLPIR is a 64-bit register.

## Field descriptions

The GICR\_INVLPIR 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
																<a href="#">pINTID</a>															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:32]

Reserved, RES0.

### pINTID, bits [31:0]

The INTID of the physical LPI to be cleaned.

---

**Note**

---

---

The size of this field is IMPLEMENTATION DEFINED, and is specified by the [GICD\\_TYPER.IDbits](#) field. Unimplemented bits are RES0.

---

**Note**

If any LPI has been forwarded to the PE and a valid write to GICR\_INVLPIR is received, the Redistributor must ensure it reloads its properties from memory and apply any changes by retrieving and reforwarding the LPI as required. This has no effect on the forwarded LPI if it has already been activated.

---

## Accessing the GICR\_INVLPIR

GICR\_INVLPIR can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	RD_base	0x00A0 - 0x00A4

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_IPRIORITYR<n>, Interrupt Priority Registers, n = 0 - 7

The GICR\_IPRIORITYR<n> characteristics are:

## Purpose

Holds the priority of the corresponding interrupt for each SGI and PPI supported by the GIC.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

These registers are used when affinity routing is enabled for the Security state of the interrupt. When affinity routing is not enabled the bits corresponding to the interrupt are RAZ/WI and [GICD\\_IPRIORITYR<n>](#) provides equivalent functionality.

These registers are used for SGIs and PPIs only. Equivalent functionality for SPIs is provided by [GICD\\_IPRIORITYR<n>](#).

These registers are byte-accessible.

When [GICD\\_CTLR](#).DS == 0:

- A field that corresponds to a Group 0 or Secure Group 1 interrupt is RAZ/WI to Non-secure accesses.
- A Non-secure access to a field that corresponds to a Non-secure Group 1 interrupt behaves as described in Software accesses of interrupt priority.

### Note

Implementations must ensure that an interrupt that is pending at the time of the write uses either the old value or the new value and must ensure that the interrupt is neither lost nor handled more than once. The effect of the change must be visible in finite time.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

A copy of these registers is provided for each Redistributor.

These registers are configured as follows:

- GICR\_IPRIORITYR0-GICR\_IPRIORITYR3 store the priority of SGIs.
- GICR\_IPRIORITYR4-GICR\_IPRIORITYR7 store the priority of PPIs.

## Attributes

GICR\_IPRIORITYR<n> is a 32-bit register.

## Field descriptions

The GICR\_IPRIORITYR<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
<a href="#">Priority_offset_3B</a>								<a href="#">Priority_offset_2B</a>								<a href="#">Priority_offset_1B</a>								<a href="#">Priority_offset_0B</a>							

**Priority\_offset\_3B, bits [31:24]**

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 3. Lower priority values correspond to greater priority of the interrupt.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Priority\_offset\_2B, bits [23:16]**

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 2. Lower priority values correspond to greater priority of the interrupt.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Priority\_offset\_1B, bits [15:8]**

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 1. Lower priority values correspond to greater priority of the interrupt.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Priority\_offset\_0B, bits [7:0]**

Interrupt priority value from an IMPLEMENTATION DEFINED range, at byte offset 0. Lower priority values correspond to greater priority of the interrupt.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Accessing the GICR\_IPRIORITYR<n>**

GICR\_IPRIORITYR<n> can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	SGI_base	$0 \times 0400 + 4n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_ISACTIVER0, Interrupt Set-Active Register 0

The GICR\_ISACTIVER0 characteristics are:

## Purpose

Activates the corresponding SGI or PPI. These registers are used when saving and restoring GIC state.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

When affinity routing is not enabled for the Security state of an interrupt in GICR\_ISACTIVER0, the corresponding bit is RAZ/WI and equivalent functionality is provided by [GICD\\_ISACTIVER<n>](#) with n=0.

This register only applies to SGIs (bits [15:0]) and PPIs (bits [31:16]). For SPIs, this functionality is provided by [GICD\\_ISACTIVER<n>](#).

When [GICD\\_CTLR.DS](#) == 0, bits corresponding to Secure SGIs and PPIs are RAZ/WI to Non-secure accesses.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_ISACTIVER0 is a 32-bit register.

## Field descriptions

The GICR\_ISACTIVER0 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
Set_active_bit<x>, bit [x], for x = 0 to 31																															

### Set\_active\_bit<x>, bit [x], for x = 0 to 31

Adds the active state to interrupt number x. Reads and writes have the following behavior:

Set_active_bit<x>	Meaning
0	If read, indicates that the corresponding interrupt is not active, and is not active and pending. If written, has no effect.
1	If read, indicates that the corresponding interrupt is active, or is active and pending. If written, activates the corresponding interrupt, if the interrupt is not already active. If the interrupt is already active, the write has no effect. After a write of 1 to this bit, a subsequent read of this bit returns 1.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICR\_ISACTIVER0

GICR\_ISACTIVER0 can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	SGI_base	0x0300

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# GICR\_ISENABLER0, Interrupt Set-Enable Register 0

The GICR\_ISENABLER0 characteristics are:

## Purpose

Enables forwarding of the corresponding SGI or PPI to the CPU interfaces.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

When affinity routing is not enabled for the Security state of an interrupt in GICR\_ISENABLER0, the corresponding bit is RAZ/WI and equivalent functionality is provided by [GICD\\_ISENABLER<n>](#) with n=0.

This register only applies to SGIs (bits [15:0]) and PPIs (bits [31:16]). For SPIs, this functionality is provided by [GICD\\_ISENABLER<n>](#).

When [GICD\\_CTLR.DS](#) == 0, bits corresponding to Secure SGIs and PPIs are RAZ/WI to Non-secure accesses.

## Configuration

Some or all RW fields of this register have defined reset values.

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_ISENABLER0 is a 32-bit register.

## Field descriptions

The GICR\_ISENABLER0 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
Set_enable_bit<x>, bit [x], for x = 0 to 31																															

### Set\_enable\_bit<x>, bit [x], for x = 0 to 31

For PPIs and SGIs, controls the forwarding of interrupt number x to the CPU interface. Reads and writes have the following behavior:

Set_enable_bit<x>	Meaning
0	If read, indicates that forwarding of the corresponding interrupt is disabled. If written, has no effect.
1	If read, indicates that forwarding of the corresponding interrupt is enabled. If written, enables forwarding of the corresponding interrupt. After a write of 1 to this bit, a subsequent read of this bit returns 1.

When this register has an architecturally-defined reset value, this field resets to 0.

# Accessing the GICR\_ISENABLER0

GICR\_ISENABLER0 can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	SGI_base	0x0100

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_ISPENDR0, Interrupt Set-Pending Register 0

The GICR\_ISPENDR0 characteristics are:

## Purpose

Adds the pending state to the corresponding SGI or PPI.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

When affinity routing is not enabled for the Security state of an interrupt in GICR\_ISPENDR0, the corresponding bit is RAZ/WI and equivalent functionality is provided by [GICD\\_ISPENDR<n>](#) with n=0.

This register only applies to SGIs (bits [15:0]) and PPIs (bits [31:16]). For SPIs, this functionality is provided by [GICD\\_ISPENDR<n>](#).

When [GICD\\_CTLR](#).DS == 0, bits corresponding to Secure SGIs and PPIs are RAZ/WI to Non-secure accesses.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_ISPENDR0 is a 32-bit register.

## Field descriptions

The GICR\_ISPENDR0 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
<a href="#">Set_pending_bit&lt;x&gt;, bit [x], for x = 0 to 31</a>																															

### Set\_pending\_bit<x>, bit [x], for x = 0 to 31

For PPIs and SGIs, adds the pending state to interrupt number x. Reads and writes have the following behavior:

Set_pending_bit<x>	Meaning
0	If read, indicates that the corresponding interrupt is not pending on this PE. If written, has no effect.
1	If read, indicates that the corresponding interrupt is pending, or active and pending on this PE. If written, changes the state of the corresponding interrupt from inactive to pending, or from active to active and pending. This has no effect in the following cases: <ul style="list-style-type: none"> <li>• If the interrupt is already pending because of a write to <a href="#">GICR_ISPENDR0</a>.</li> <li>• If the interrupt is already pending because the corresponding interrupt signal is asserted. In this case, the interrupt remains pending if the interrupt signal is deasserted.</li> </ul>

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICR\_ISPENDR0

GICR\_ISPENDR0 can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	SGI_base	0x0200

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_NSACR, Non-secure Access Control Register

The GICR\_NSACR characteristics are:

## Purpose

Enables Secure software to permit Non-secure software to create SGIs targeting the PE connected to this Redistributor by writing to [ICC\\_SGI1R\\_EL1](#), [ICC\\_ASGI1R\\_EL1](#) or [ICC\\_SGI0R\\_EL1](#).

See Forwarding an SGI to a target PE for more information.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

When [GICD\\_CTLR](#).DS == 1, this register is RAZ/WI.

When [GICD\\_CTLR](#).DS == 0, this register is Secure, and is RAZ/WI to Non-secure accesses.

This register is used when affinity routing is enabled. When affinity routing is not enabled for the Security state of the interrupt, [GICD\\_NSACR<n>](#) with n=0 provides equivalent functionality.

This register does not support PPIs.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

For a description on when a write to [ICC\\_SGI0R\\_EL1](#), [ICC\\_SGI1R\\_EL1](#) or [ICC\\_ASGI1R\\_EL1](#) is permitted to generate an interrupt see Use of control registers for SGI forwarding.

## Attributes

GICR\_NSACR is a 32-bit register.

## Field descriptions

The GICR\_NSACR 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
NS_access<x>, bits [2x+1:2x], for x = 0 to 15																															

**NS\_access<x>, bits [2x+1:2x], for x = 0 to 15**

Configures the level of Non-secure access permitted when the SGI is in Secure Group 0 or Secure Group 1, as defined from [GICR\\_IGROUPR0](#) and [GICR\\_IGRPMODR0](#). A field is provided for each SGI. The possible values of each 2-bit field are:

NS_access<x>	Meaning
00	Non-secure writes are not permitted to generate Secure Group 0 SGIs or Secure Group 1 SGIs.
01	Non-secure writes are permitted to generate a Secure Group 0 SGI.
10	As 0b01, but additionally Non-secure writes to are permitted to generate a Secure Group 1 SGI.
11	Reserved. If the field is programmed to the reserved value, then the hardware will treat the field as if it has been programmed to an IMPLEMENTATION DEFINED choice of the valid values. However, to maintain the principle that as the value increases additional accesses are permitted ARM strongly recommends that implementations treat this value as 10. It is IMPLEMENTATION DEFINED whether the value read back is the value programmed or the valid value chosen.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICR\_NSACR

GICR\_NSACR can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	SGI_base	0x0E00

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_PENDBASER, Redistributor LPI Pending Table Base Address Register

The GICR\_PENDBASER characteristics are:

## Purpose

Specifies the base address of the LPI Pending table, and the Shareability and Cacheability of accesses to the LPI Pending table.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

Having the GICR\_PENDBASER OuterCache, Shareability or InnerCache fields programmed to different values on different Redistributors with [GICR\\_CTLR.EnableLPIs](#) == 1 in the system is UNPREDICTABLE.

Changing GICR\_PENDBASER with [GICR\\_CTLR.EnableLPIs](#) == 1 is UNPREDICTABLE.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_PENDBASER is a 64-bit register.

## Field descriptions

The GICR\_PENDBASER 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	PTZ	0	0	0	OuterCache	0	0	0	0	Physical_Address																					
Physical_Address															0	0	0	0	Shareability	InnerCache	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

### Bit [63]

Reserved, RES0.

### PTZ, bit [62]

Pending Table Zero. Indicates to the Redistributor whether the LPI Pending table is zero when [GICR\\_CTLR.EnableLPIs](#) == 1.

This field is WO, and reads as 0.

PTZ	Meaning
0	The LPI Pending table is not zero, and contains live data.
1	The LPI Pending table is zero. Software must ensure the LPI Pending table is zero before this value is written.

**Bits [61:59]**

Reserved, RES0.

**OuterCache, bits [58:56]**

Indicates the Outer Cacheability attributes of accesses to the LPI Pending table. The possible values of this field are:

OuterCache	Meaning
000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
001	Normal Outer Non-cacheable.
010	Normal Outer Cacheable Read-allocate, Write-through.
011	Normal Outer Cacheable Read-allocate, Write-back.
100	Normal Outer Cacheable Write-allocate, Write-through.
101	Normal Outer Cacheable Write-allocate, Write-back.
110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

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 [55:52]**

Reserved, RES0.

**Physical\_Address, bits [51:16]**

Bits [51:16] of the physical address containing the LPI Pending table.

In implementations supporting fewer than 52 bits of physical address, unimplemented upper bits are RES0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Bits [15:12]**

Reserved, RES0.

**Shareability, bits [11:10]**

Indicates the Shareability attributes of accesses to the LPI Pending table. The possible values of this field are:

Shareability	Meaning
00	Non-shareable.
01	Inner Shareable.
10	Outer Shareable.
11	Reserved. Treated as 00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

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.



**InnerCache, bits [9:7]**

Indicates the Inner Cacheability attributes of accesses to the LPI Pending table. The possible values of this field are:

InnerCache	Meaning
000	Device-nGnRnE.
001	Normal Inner Non-cacheable.
010	Normal Inner Cacheable Read-allocate, Write-through.
011	Normal Inner Cacheable Read-allocate, Write-back.
100	Normal Inner Cacheable Write-allocate, Write-through.
101	Normal Inner Cacheable Write-allocate, Write-back.
110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Bits [6:0]**

Reserved, RES0.

**Accessing the GICR\_PENDBASER**

GICR\_PENDBASER can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	RD_base	0x0078 - 0x007C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_PROPBASER, Redistributor Properties Base Address Register

The GICR\_PROPBASER characteristics are:

## Purpose

Specifies the base address of the LPI Configuration table, and the Shareability and Cacheability of accesses to the LPI Configuration table.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

It is IMPLEMENTATION DEFINED whether GICR\_PROPBASER can be set to different values on different Redistributors. [GICR\\_TYPER.CommonLPIAff](#) identifies the Redistributors that must have GICR\_PROPBASER set to the same values whenever [GICR\\_CTLR.EnableLPIs](#) == 1.

Setting different values in different copies of GICR\_PROPBASER on Redistributors that are required to use a common LPI Configuration table when [GICR\\_CTLR.EnableLPIs](#) == 1 leads to UNPREDICTABLE behavior.

Other restrictions apply when a Redistributor caches information from GICR\_PROPBASER. See LPI Configuration tables for more information.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

A copy of this register is provided for each Redistributor.

An implementation might make this register RO, for example to correspond to an LPI Configuration table in read-only memory.

## Attributes

GICR\_PROPBASER is a 64-bit register.

## Field descriptions

The GICR\_PROPBASER 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	OuterCache	0	0	0	0	Physical_Address																							
Physical_Address												Shareability				InnerCache		0	0	IDbits													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		

### Bits [63:59]

Reserved, RES0.

### OuterCache, bits [58:56]

Indicates the Outer Cacheability attributes of accesses to the LPI Configuration table. The possible values of this field are:

OuterCache	Meaning
000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
001	Normal Outer Non-cacheable.
010	Normal Outer Cacheable Read-allocate, Write-through.
011	Normal Outer Cacheable Read-allocate, Write-back.
100	Normal Outer Cacheable Write-allocate, Write-through.
101	Normal Outer Cacheable Write-allocate, Write-back.
110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

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 [55:52]

Reserved, RES0.

#### Physical\_Address, bits [51:12]

Bits [51:12] of the physical address containing the LPI Configuration table.

In implementations supporting fewer than 52 bits of physical address, unimplemented upper bits are RES0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Shareability, bits [11:10]

Indicates the Shareability attributes of accesses to the LPI Configuration table. The possible values of this field are:

Shareability	Meaning
00	Non-shareable.
01	Inner Shareable.
10	Outer Shareable.
11	Reserved. Treated as 00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

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.

#### InnerCache, bits [9:7]

Indicates the Inner Cacheability attributes of accesses to the LPI Configuration table. The possible values of this field are:

InnerCache	Meaning
000	Device-nGnRnE.
001	Normal Inner Non-cacheable.
010	Normal Inner Cacheable Read-allocate, Write-through.
011	Normal Inner Cacheable Read-allocate, Write-back.
100	Normal Inner Cacheable Write-allocate, Write-through.
101	Normal Inner Cacheable Write-allocate, Write-back.
110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Bits [6:5]

Reserved, RES0.

**IDbits, bits [4:0]**

The number of bits of LPI INTID supported, minus one, by the LPI Configuration table starting at Physical\_Address.

If the value of this field is larger than the value of [GICD\\_TYPER.IDbits](#), the [GICD\\_TYPER.IDbits](#) value applies.

If the value of this field is less than 0b1101, indicating that the largest INTID is less than 8192 (the smallest LPI interrupt ID), the GIC will behave as if all physical LPIs are out of range.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Accessing the GICR\_PROPBASER**

GICR\_PROPBASER can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	RD_base	0x0070 - 0x0074

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_SETLPIR, Set LPI Pending Register

The GICR\_SETLPIR characteristics are:

## Purpose

Generates an LPI by setting the pending state of the specified LPI.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

When written with a 32-bit write the data is zero-extended to 64 bits.

This register is mandatory in an implementation that supports LPIs and does not include an ITS. The functionality is IMPLEMENTATION DEFINED in an implementation that does include an ITS.

Writes to this register have no effect if either:

- The pINTID field corresponds to an LPI that is already pending.
- The pINTID field corresponds to an unimplemented LPI.
- [GICR\\_CTLR.EnableLPIs](#) = 0.

## Configuration

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_SETLPIR is a 64-bit register.

## Field descriptions

The GICR\_SETLPIR 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
																pINTID															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:32]

Reserved, RES0.

### pINTID, bits [31:0]

The INTID of the physical LPI to be generated.

---

#### Note

---

---

The size of this field is IMPLEMENTATION DEFINED, and is specified by the [GICD\\_TYPER.IDbits](#) field. Unimplemented bits are RES0.

---

## Accessing the GICR\_SETLPIR

GICR\_SETLPIR can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	RD_base	0x0040 - 0x0044

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_STATUSR, Error Reporting Status Register

The GICR\_STATUSR characteristics are:

## Purpose

Provides software with a mechanism to detect:

- Accesses to reserved locations.
- Writes to read-only locations.
- Reads of write-only locations.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

In an implementation that supports two Security states, there are separate Secure and Non-secure instances of this register:

	Security disabled	Secure	Non-secure
GICR_STATUSR(S)	RW	RW	-
GICR_STATUSR(NS)	RW	-	RW

This is an optional register. If the register is not implemented, the location is RAZ/WI.

## Configuration

A copy of this register is provided for each Redistributor.

If the GIC implementation supports two Security states this register is Banked to provide Secure and Non-secure copies.

## Attributes

GICR\_STATUSR is a 32-bit register.

## Field descriptions

The GICR\_STATUSR 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	WROD	RWOD	WRD	RRD

### Bits [31:4]

Reserved, RES0.

### WROD, bit [3]

Write to an RO location.

WROD	Meaning
0	Normal operation.
1	A write to an RO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**RWOD, bit [2]**

Read of a WO location.

<b>RWOD</b>	<b>Meaning</b>
0	Normal operation.
1	A read of a WO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**WRD, bit [1]**

Write to a reserved location.

<b>WRD</b>	<b>Meaning</b>
0	Normal operation.
1	A write to a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**RRD, bit [0]**

Read of a reserved location.

<b>RRD</b>	<b>Meaning</b>
0	Normal operation.
1	A read of a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**Accessing the GICR\_STATUSR**

GICR\_STATUSR can be accessed through its memory-mapped interface:

<b>Component</b>	<b>Frame</b>	<b>Offset</b>
GIC Redistributor	RD_base	0x0010



# GICR\_SYNCNCR, Redistributor Synchronize Register

The GICR\_SYNCNCR characteristics are:

## Purpose

Indicates completion of physical Redistributor operations.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

Optionally, when this register is accessed, an implementation might wait until all operations are complete before returning a value, in which case GICR\_SYNCNCR.Busy is always 0.

This register is mandatory in an implementation that supports LPIs and does not include an ITS. The functionality is IMPLEMENTATION DEFINED in an implementation that does include an ITS.

## Configuration

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_SYNCNCR is a 32-bit register.

## Field descriptions

The GICR\_SYNCNCR 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	Busy

### Bits [31:1]

Reserved, RES0.

### Busy, bit [0]

Indicates completion of any Redistributor operations as follows:

Busy	Meaning
0	No operations are in progress.
1	A write is in progress to one or more of the following registers: <ul style="list-style-type: none"><li><a href="#">GICR_CLRLPIR</a>.</li><li><a href="#">GICR_INVLPPIR</a>.</li><li><a href="#">GICR_INVALLR</a>.</li></ul>

This field also indicates completion of any operations initiated by writes to [GICR\\_PENDBASER](#) or [GICR\\_PROPBASER](#).

# Accessing the GICR\_SYNCNR

GICR\_SYNCNR can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	RD_base	0x00C0 - 0x00C4

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_TYPER, Redistributor Type Register

The GICR\_TYPER characteristics are:

## Purpose

Provides information about the configuration of this Redistributor.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

## Configuration

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_TYPER is a 64-bit register.

## Field descriptions

The GICR\_TYPER 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						
Affinity_Value																																					
0	0	0	0	0	0	CommonLPIAff										Processor_Number										0	0	DPGS		Last	DirectLPI		0	VLPIS		PLPIS	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						

### Affinity\_Value, bits [63:32]

The identity of the PE associated with this Redistributor.

Bits [63:56] provide Aff3, the Affinity level 3 value for the Redistributor.

Bits [55:48] provide Aff2, the Affinity level 2 value for the Redistributor.

Bits [47:40] provide Aff1, the Affinity level 1 value for the Redistributor.

Bits [39:32] provide Aff0, the Affinity level 0 value for the Redistributor.

### Bits [31:26]

Reserved, RES0.

### CommonLPIAff, bits [25:24]

The affinity level at which Redistributors share a LPI Configuration table.

CommonLPIAff	Meaning
00	All Redistributors must share an LPI Configuration table.
01	All Redistributors with the same Aff3 value must share an LPI Configuration table.
10	All Redistributors with the same Aff3.Aff2 value must share an LPI Configuration table.
11	All Redistributors with the same Aff3.Aff2.Aff1 value must share an LPI Configuration table.

**Processor\_Number, bits [23:8]**

A unique identifier for the PE. When [GITS\\_TYPER](#).PTA == 0, an ITS uses this field to identify the interrupt target.

When affinity routing is disabled for a Security state, this field indicates which [GICD\\_ITARGETSR<n>](#) corresponds to this Redistributor.

**Bits [7:6]**

Reserved, RES0.

**DPGS, bit [5]**

Sets support for [GICR\\_CTLR](#).DPG\* bits.

DPGS	Meaning
0	<a href="#">GICR_CTLR</a> .DPG* bits are not supported.
1	<a href="#">GICR_CTLR</a> .DPG* bits are supported.

**Last, bit [4]**

Indicates whether this Redistributor is the highest-numbered Redistributor in a series of contiguous Redistributor pages.

Last	Meaning
0	This Redistributor is not the highest-numbered Redistributor in a series of contiguous Redistributor pages.
1	This Redistributor is the highest-numbered Redistributor in a series of contiguous Redistributor pages.

**DirectLPI, bit [3]**

Indicates whether this Redistributor supports direct injection of LPIs.

DirectLPI	Meaning
0	This Redistributor does not support direct injection of LPIs. The <a href="#">GICR_SETLPIR</a> , <a href="#">GICR_CLRLPIR</a> , <a href="#">GICR_INVLPIR</a> , <a href="#">GICR_INVALLR</a> , and <a href="#">GICR_SYNCRR</a> registers are either not implemented, or have an IMPLEMENTATION DEFINED purpose.
1	This Redistributor supports direct injection of LPIs. The <a href="#">GICR_SETLPIR</a> , <a href="#">GICR_CLRLPIR</a> , <a href="#">GICR_INVLPIR</a> , <a href="#">GICR_INVALLR</a> , and <a href="#">GICR_SYNCRR</a> registers are implemented.

**Bit [2]**

Reserved, RES0.

**VLPIS, bit [1]**

Indicates whether the GIC implementation supports virtual LPIs and the direct injection of virtual LPIs.

VLPIS	Meaning
0	The implementation does not support virtual LPIs or the direct injection of virtual LPIs.
1	The implementation supports virtual LPIs and the direct injection of virtual LPIs.

---

**Note**

In GICv3 implementations this field is RES0.

---

**PLPIS, bit [0]**

Indicates whether the GIC implementation supports physical LPis.

PLPIS	Meaning
0	The implementation does not support physical LPis.
1	The implementation supports physical LPis.

## Accessing the GICR\_TYPER

GICR\_TYPER can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	RD_base	0x0008 - 0x000C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_VPENDBASER, Virtual Redistributor LPI Pending Table Base Address Register

The GICR\_VPENDBASER characteristics are:

## Purpose

Specifies the base address of the memory that holds the virtual LPI Pending table for the currently scheduled virtual machine.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

The effect of a write to this register is not guaranteed to be visible throughout the affinity hierarchy, as indicated by [GICR\\_CTLR](#).RWP == 0.

## Configuration

Some or all RW fields of this register have defined reset values.

This register is provided only in GICv4 implementations.

## Attributes

GICR\_VPENDBASER is a 64-bit register.

## Field descriptions

The GICR\_VPENDBASER 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								
Valid	IDA	PendingLast	Dirty	0	OuterCache	0	0	0	0	Physical_Address																													
Physical_Address														0	0	0	0	Shareability	InnerCache	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								

### Valid, bit [63]

This bit controls whether the virtual LPI Pending table is valid:

Valid	Meaning
0	The virtual LPI Pending table is not valid. No vPE is scheduled.
1	The virtual LPI Pending table is valid. A vPE is scheduled.

Setting GICR\_VPENDBASER.Valid == 1 when the associated CPU interface does not implement GICv4 is UNPREDICTABLE.

### Note

Software can determine whether a PE supports GICv3 or GICv4 by reading ID\_AA64PFR0\_EL1.

Writing a new value to any bit of GICR\_VPENDBASER, other than GICR\_VPENDBASER.Valid, when GICR\_VPENDBASER.Valid==1 is UNPREDICTABLE.

When this register has an architecturally-defined reset value, this field resets to 0.

### IDAI, bit [62]

Implementation Defined Area Invalid. Indicates whether the IMPLEMENTATION DEFINED area in the virtual LPI Pending table is valid:

IDAI	Meaning
0	The IMPLEMENTATION DEFINED area is valid.
1	The IMPLEMENTATION DEFINED area is invalid and all pending interrupt information is held in the architecturally defined part of the virtual LPI Pending table.

For more information, see LPI Pending tables and Virtual LPI Configuration tables and virtual LPI Pending tables.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### PendingLast, bit [61]

Indicates whether there are pending and enabled interrupts for the last scheduled vPE.

This value is set by the implementation when GICR\_VPENDBASER.Valid has been written from 1 to 0 and is otherwise UNKNOWN.

PendingLast	Meaning
0	There are no pending and enabled interrupts for the last scheduled vPE.
1	There is at least one pending interrupt for the last scheduled vPE. It is IMPLEMENTATION DEFINED whether this bit is set when the only pending interrupts for the last scheduled vPE are not enabled. ARM deprecates setting PendingLast to 1 when the only pending interrupts for the last scheduled virtual machine are not enabled.

When the GICR\_VPENDBASER.Valid bit is written from 0 to 1, then the state of this bit indicates to the hardware whether the virtual LPI Pending table contains no pending interrupts:

- 0b0: The virtual LPI Pending table is known to be zero, and so the virtual LPI Pending table does not need to be read by hardware to determine which pending interrupts are present.
- 0b1: The virtual LPI Pending table is not known to be zero, and so the hardware must read the virtual LPI Pending table to determine which pending interrupts are present.

When this register has an architecturally-defined reset value, this field resets to 0.

### Dirty, bit [60]

Read-only. Indicates whether there are any virtual LPIs for the last scheduled vPE that have not completed. This field is used only when GICR\_VPENDBASER.Valid==0, and is otherwise UNKNOWN:

Dirty	Meaning
0	There are no uncompleted virtual LPIs for the last scheduled vPE.
1	There is at least one uncompleted virtual LPI for the last scheduled vPE.

#### Note

When GICR\_VPENDBASER.Valid == 0, the Redistributor must ensure any outstanding pending virtual interrupts are cleared from the CPU interface.

Writing to GICR\_VPENDBASER when GICR\_VPENDBASER.Dirty==1 is UNPREDICTABLE.

When this register has an architecturally-defined reset value, this field resets to 0.

### Bit [59]

Reserved, RES0.

**OuterCache, bits [58:56]**

Indicates the Outer Cacheability attributes of accesses to virtual LPI Pending tables of vPEs targeting this Redistributor. The possible values of this field are:

OuterCache	Meaning
000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
001	Normal Outer Non-cacheable.
010	Normal Outer Cacheable Read-allocate, Write-through.
011	Normal Outer Cacheable Read-allocate, Write-back.
100	Normal Outer Cacheable Write-allocate, Write-through.
101	Normal Outer Cacheable Write-allocate, Write-back.
110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The Cacheability, Outer Cacheability and Shareability fields are used for accesses to the virtual LPI Pending table of resident and non-resident vPEs.

If the OuterCacheability attribute of the virtual LPI Pending tables that are associated with vPEs targeting the same Redistributor are different, behavior is UNPREDICTABLE.

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 [55:52]**

Reserved, RES0.

**Physical\_Address, bits [51:16]**

Bits [51:16] of the physical address containing the virtual LPI Pending table.

In implementations supporting fewer than 52 bits of physical address, unimplemented upper bits are RES0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Bits [15:12]**

Reserved, RES0.

**Shareability, bits [11:10]**

Indicates the Shareability attributes of accesses to the virtual LPI Pending table. The possible values of this field are:

Shareability	Meaning
00	Non-shareable.
01	Inner Shareable.
10	Outer Shareable.
11	Reserved. Treated as 00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

The Cacheability, Outer Cacheability and Shareability fields are used for accesses to the virtual LPI Pending table of resident and non-resident vPEs.

If the Shareability attribute of the virtual LPI Pending tables that are associated with vPEs targeting the same Redistributor are different, behavior is UNPREDICTABLE.

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.



**InnerCache, bits [9:7]**

Indicates the Inner Cacheability attributes of accesses to the virtual LPI Pending table. The possible values of this field are:

InnerCache	Meaning
000	Device-nGnRnE.
001	Normal Inner Non-cacheable.
010	Normal Inner Cacheable Read-allocate, Write-through.
011	Normal Inner Cacheable Read-allocate, Write-back.
100	Normal Inner Cacheable Write-allocate, Write-through.
101	Normal Inner Cacheable Write-allocate, Write-back.
110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

The Cacheability, Outer Cacheability and Shareability fields are used for accesses to the virtual LPI Pending table of resident and non-resident vPEs.

If the InnerCacheability attribute of the virtual LPI Pending tables that are associated with vPEs targeting the same Redistributor are different, behavior is UNPREDICTABLE.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Bits [6:0]**

Reserved, RES0.

**Accessing the GICR\_VPENDBASER**

GICR\_VPENDBASER can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	VLPI_base	0x0078 - 0x007C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICR\_VPROPBASER, Virtual Redistributor Properties Base Address Register

The GICR\_VPROPBASER characteristics are:

## Purpose

Specifies the base address of the memory that holds the virtual LPI Configuration table for the currently scheduled virtual machine.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

This register is provided in GICv4 implementations only.

## Attributes

GICR\_VPROPBASER is a 64-bit register.

## Field descriptions

The GICR\_VPROPBASER 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	OuterCache	0	0	0	0	Physical_Address																					
Physical_Address											Shareability		InnerCache		0	0	IDbits														
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:59]

Reserved, RES0.

### OuterCache, bits [58:56]

Indicates the Outer Cacheability attributes of accesses to the LPI Configuration table. The possible values of this field are:

OuterCache	Meaning
000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
001	Normal Outer Non-cacheable.
010	Normal Outer Cacheable Read-allocate, Write-through.
011	Normal Outer Cacheable Read-allocate, Write-back.
100	Normal Outer Cacheable Write-allocate, Write-through.
101	Normal Outer Cacheable Write-allocate, Write-back.
110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

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 [55:52]

Reserved, RES0.

#### Physical\_Address, bits [51:12]

Bits [51:12] of the physical address containing the virtual LPI Configuration table.

In implementations supporting fewer than 52 bits of physical address, unimplemented upper bits are RES0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Shareability, bits [11:10]

Indicates the Shareability attributes of accesses to the LPI Configuration table. The possible values of this field are:

Shareability	Meaning
00	Non-shareable.
01	Inner Shareable.
10	Outer Shareable.
11	Reserved. Treated as 00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

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.

#### InnerCache, bits [9:7]

Indicates the Inner Cacheability attributes of accesses to the LPI Configuration table. The possible values of this field are:

InnerCache	Meaning
000	Device-nGnRnE.
001	Normal Inner Non-cacheable.
010	Normal Inner Cacheable Read-allocate, Write-through.
011	Normal Inner Cacheable Read-allocate, Write-back.
100	Normal Inner Cacheable Write-allocate, Write-through.
101	Normal Inner Cacheable Write-allocate, Write-back.
110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Bits [6:5]

Reserved, RES0.

#### IDbits, bits [4:0]

The number of bits of virtual LPI INTID supported, minus one.

If the value of this field is less than 0b1101, indicating that the largest INTID is less than 8192 (the smallest LPI interrupt ID), the GIC will behave as if all virtual LPIs are out of range.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICR\_VPROPBASER

GICR\_VPROPBASER can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	VLPI_base	0x0070 - 0x0074

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

## GICR\_WAKER, Redistributor Wake Register

The GICR\_WAKER characteristics are:

## Purpose

Permits software to control the behavior of the **WakeRequest** power management signal corresponding to the Redistributor. Power management operations follow the rules in Power management.

This register is part of the GIC Redistributor registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RAZ/WI

When `GICD_CTLR.DS==1`, this register is always accessible.

When [GICD\\_CTLR.DS==0](#), this is a Secure register. This register is RAZ/WI to Non-secure accesses.

To ensure a Redistributor is quiescent, software must write to GICR\_WAKER with ProcessorSleep == 1, then poll the register until ChildrenAsleep == 1.

Resetting the connected PE when GICR\_WAKER.ProcessorSleep==0 or GICR\_WAKER.ChildresAsleep==0, can lead to UNPREDICTABLE behaviour in the IRI.

Resetting the IRI when GICR\_WAKER.ProcessorSleep==0 or GICR\_WAKER.ChildresAsleep==0 can lead to UNPREDICTABLE behaviour in the connected PE.

## Configuration

Some or all RW fields of this register have defined reset values.

A copy of this register is provided for each Redistributor.

## Attributes

GICR\_WAKER is a 32-bit register.

## Field descriptions

The GICR WAKER bit assignments are:

[illegible]

**IMPLEMENTATION DEFINED, bit [31]**

## IMPLEMENTATION DEFINED.

**Bits [30:3]**

Reserved, RES0.

**ChildrenAsleep, bit [2]**

Read-only. Indicates whether the connected PE is quiescent:

ChildrenAsleep	Meaning
0	An interface to the connected PE might be active.
1	All interfaces to the connected PE are quiescent.

When this register has an architecturally-defined reset value, this field resets to 1.

**ProcessorSleep, bit [1]**

Indicates whether the Redistributor can assert the **WakeRequest** signal:

ProcessorSleep	Meaning
0	This PE is not in, and is not entering, a low power state.
1	The PE is either in, or is in the process of entering, a low power state. All interrupts that arrive at the Redistributor: <ul style="list-style-type: none"> <li>• Assert a <b>WakeRequest</b> signal.</li> <li>• Are held in the pending state at the Redistributor, and are not communicated to the CPU interface.</li> </ul>

**Note**

When ProcessorSleep == 1, the Redistributor must ensure that any interrupts that are pending on the CPU interface are released.

For an implementation that is using the GIC Stream Protocol Interface:

- A Quiesce command can put the interface between the Redistributor and the CPU interface in a quiescent state.
- A Release command can release any interrupts that are pending on the CPU interface.

**Note**

Before powering down a PE, software must set this bit to 1 and wait until ChildrenAsleep == 1.  
After powering up a PE, or following a failed powerdown, software must set this bit to 0 and wait until ChildrenAsleep == 0.

Changing ProcessorSleep from 1 to 0 when ChildrenAsleep is not 1 results in UNPREDICTABLE behavior.

Changing ProcessorSleep from 0 to 1 when the Enable for each interrupt group in the associated CPU interface is not 0 results in UNPREDICTABLE behavior.

When this register has an architecturally-defined reset value, this field resets to 1.

**IMPLEMENTATION DEFINED, bit [0]**

IMPLEMENTATION DEFINED.

**Accessing the GICR\_WAKER**

GICR\_WAKER can be accessed through its memory-mapped interface:

Component	Frame	Offset
GIC Redistributor	RD_base	0x0014

# GICV\_ABPR, Virtual Machine Aliased Binary Point Register

The GICV\_ABPR characteristics are:

## Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption.

This register corresponds to [GICC\\_ABPR](#) in the physical CPU interface.

---

### Note

[GICH\\_LR<n>](#).Group determines whether a virtual interrupt is Group 0 or Group 1.

---

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_BPR1](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_BPR1\\_EL1](#) provides equivalent functionality.

The value contained in this register is one greater than the actual applied binary point value, as described in 'Priority grouping' in the GICv3 Architecture Specification.

This register is used for Group 1 interrupts when [GICV\\_CTLR](#).CBPR == 0. [GICV\\_BPR](#) provides equivalent functionality for Group 0 interrupts, and for Group 1 interrupts when [GICV\\_CTLR](#).CBPR == 1.

## Configuration

Some or all RW fields of this register have defined reset values.

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICV\_ABPR is a 32-bit register.

## Field descriptions

The GICV\_ABPR 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	Binary_Point

### Bits [31:3]

Reserved, RES0.

**Binary\_Point, bits [2:0]**

Controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field.

For information about how this field determines the interrupt priority bits assigned to the group priority field, see Priority grouping.

When this register has an architecturally-defined reset value, this field resets to 0.

The Binary\_Point field of this register is aliased to [GICH\\_VMCR.VBPR1](#).

## Accessing the GICV\_ABPR

GICV\_ABPR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual CPU interface	0x001C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# GICV\_AEOIR, Virtual Machine Aliased End Of Interrupt Register

The GICV\_AEOIR characteristics are:

## Purpose

A write to this register performs a priority drop for the specified Group 1 virtual interrupt and, if [GICV\\_CTLR.EOImode](#) == 0, also deactivates the interrupt.

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_EOIR1](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_EOIR1\\_EL1](#) provides equivalent functionality.

This register is used for Group 1 interrupts only. [GICV\\_EOIR](#) provides equivalent functionality for Group 0 interrupts.

When affinity routing is enabled, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICV\_AEOIR is a 32-bit register.

## Field descriptions

The GICV\_AEOIR 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	INTID																										

### Bits [31:25]

Reserved, RES0.

### INTID, bits [24:0]

The INTID of the signaled interrupt.

---

#### Note

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

---

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

A successful EOI request means that:

- The highest priority bit in [GICH\\_APR<n>](#) is cleared, causing the running priority to drop.
- If the appropriate [GICV\\_CTLR](#).EOImode bit == 0, the interrupt is deactivated in the corresponding List register. If the INTID corresponds to a hardware interrupt, the interrupt is also deactivated in the Distributor.

---

#### Note

Only Group 1 interrupts can target the hypervisor, and therefore only Group 1 interrupts are deactivated in the Distributor.

---

A write to this register is UNPREDICTABLE if the INTID corresponds to a Group 0 interrupt. In addition, the following GICv2 UNPREDICTABLE cases require specific actions:

- If highest active priority is Group 0 and the identified interrupt is in the List Registers and it matches the highest active priority. When EL2 is using System registers and [ICH\\_VTR\\_EL2](#).SEIS is 1, an IMPLEMENTATION DEFINED SEI might be generated, otherwise GICv3 implementations must ignore such writes.
- If the identified interrupt is in the List Registers, and the HW bit is 1, and the interrupt to be deactivated is an SGI (that is, the value of [Physical\\_ID](#) is between 0 and 15). GICv3 implementations must perform the deactivate operation. This means that a GICv3 implementation in legacy operation must ensure only a single SGI is active for a PE.
- If the identified interrupt is in the List Registers, and the HW bit is 1, and the corresponding pINTID field value is between 1020 and 1023, indicating a special purpose INTID. GICv3 implementations must not perform a deactivate operation but must still change the state of the List register as appropriate. When EL2 is using System registers and [ICH\\_VTR\\_EL2](#).SEIS is 1, an implementation might generate a system error.

## Accessing the GICV\_AEOIR

GICV\_AEOIR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual CPU interface	0x0024

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICV\_AHPPIR, Virtual Machine Aliased Highest Priority Pending Interrupt Register

The GICV\_AHPPIR characteristics are:

## Purpose

Provides the INTID of the highest priority pending Group 1 virtual interrupt in the List registers.

This register corresponds to the physical CPU interface register [GICC\\_AHPPIR](#).

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_HPIR1](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_HPIR1\\_EL1](#) provides equivalent functionality.

This register is used for Group 1 interrupts only. [GICV\\_HPIR](#) provides equivalent functionality for Group 0 interrupts.

The register does not return the INTID of an interrupt that is active and pending.

When affinity routing is enabled, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICV\_AHPPIR is a 32-bit register.

## Field descriptions

The GICV\_AHPPIR 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																									

### Bits [31:25]

Reserved, RES0.

### INTID, bits [24:0]

The INTID of the signaled interrupt.

---

**Note**

---

---

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

---

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

A read of this register returns the spurious INTID 1023 if any of the following are true:

- There are no pending interrupts of sufficiently high priority value to be signaled to the PE.
- The highest priority pending interrupt is in Group 0.

## Accessing the GICV\_AHPPIR

GICV\_AHPPIR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual CPU interface	0x0028

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

## GICV\_AIAR, Virtual Machine Aliased Interrupt Acknowledge Register

The GICV\_AIAR characteristics are:

## Purpose

Provides the INTID of the signaled Group 1 virtual interrupt. A read of this register by the PE acts as an acknowledge for the interrupt.

This register corresponds to the physical CPU interface register [GICC\\_AIAR](#).

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_IAR1](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_IAR1\\_EL1](#) provides equivalent functionality.

This register is used for Group 1 interrupts only. [GICV\\_IAR](#) provides equivalent functionality for Group 0 interrupts.

When affinity routing is enabled, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICV AIAR is a 32-bit register.

## Field descriptions

The GICV AIAR bit assignments are:

[illegible]

**Bits [31:25]**

Reserved, RES0.

**INTID, bits [24:0]**

The INTID of the signaled interrupt.

### Note

---

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

---

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

The operation of this register is similar to the operation of [GICV\\_IAR](#). When a vPE reads this register, the corresponding [GICH\\_LR<n>](#).Group field is checked to determine whether the interrupt is in Group 0 or Group 1:

- If the interrupt is Group 0, the spurious INTID 1023 is returned and the interrupt is not acknowledged.
- If the interrupt is Group 1, the INTID is returned. The List register entry is updated to active state, and the appropriate bit in [GICH\\_APR<n>](#) is set to 1.

A read of this register returns the spurious INTID 1023 if any of the following are true:

- When the virtual CPU interface is enabled and [GICH\\_HCR](#).En == 1:
  - There are no pending interrupts of sufficiently high priority value to be signaled to the PE.
  - The highest priority pending interrupt is in Group 0.
- Interrupt signaling by the virtual CPU interface is disabled.

## Accessing the GICV\_AIAR

GICV\_AIAR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual CPU interface	0x0020

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICV\_APR<n>, Virtual Machine Active Priorities Registers, n = 0 - 3

The GICV\_APR<n> characteristics are:

## Purpose

Provides information about interrupt active priorities.

These registers correspond to the physical CPU interface registers [GICC\\_APR<n>](#).

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

If System register access is not enabled for EL2, these registers access [GICH\\_APR<n>](#). If System register access is enabled for EL2, these registers access [ICH\\_APIR<n>\\_EL2](#). All active priority mapped guests are held in the accessed registers, regardless of interrupt group.

## Configuration

When System register access is disabled for EL2, these registers access [GICH\\_APR<n>](#), and all active priorities for virtual machines are held in [GICH\\_APR<n>](#) regardless of interrupt group.

When System register access is enabled for EL2, these registers access [ICH\\_APIR<n>\\_EL2](#), and all active priorities for virtual machines are held in [ICH\\_APIR<n>\\_EL2](#) regardless of interrupt group.

## Attributes

GICV\_APR<n> is a 32-bit register.

## Field descriptions

The GICV\_APR<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
P31	P30	P29	P28	P27	P26	P25	P24	P23	P22	P21	P20	P19	P18	P17	P16	P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0

**P<x>, bit [x], for x = 0 to 31**

Provides information about active priorities for the virtual machine.

See [GICH\\_APR<n>](#) and [ICH\\_APIR<n>\\_EL2](#) for the correspondence between priorities and bits.

## Accessing the GICV\_APR<n>

GICV\_APR<n> can be accessed through its memory-mapped interface:

Component	Offset
-----------	--------

GIC Virtual CPU interface	0x00D0 + 4n
------------------------------	-------------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# GICV\_BPR, Virtual Machine Binary Point Register

The GICV\_BPR characteristics are:

## Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 0 interrupt preemption.

This register corresponds to [GICC\\_BPR](#) in the physical CPU interface.

---

### Note

[GICH\\_LR<n>](#).Group determines whether a virtual interrupt is Group 0 or Group 1.

---

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_BPR0](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_BPR0\\_EL1](#) provides equivalent functionality.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

This register is available when the GIC implementation supports interrupt virtualization.

When [GICV\\_CTLR](#).CBPR == 1, this register determines interrupt preemption for both Group 0 and Group 1 interrupts.

## Attributes

GICV\_BPR is a 32-bit register.

## Field descriptions

The GICV\_BPR 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	<a href="#">Binary_Point</a>

### Bits [31:3]

Reserved, RES0.

### Binary\_Point, bits [2:0]

Controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field.

For information about how this field determines the interrupt priority bits assigned to the group priority field, see Priority grouping for Group 0 interrupts, or Group 1 interrupts when CBPR==1

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

The Binary\_Point field of this register is aliased to [GICH\\_VMCR.VBPR0](#).

## Accessing the GICV\_BPR

GICV\_BPR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual CPU interface	0x0008

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICV\_CTLR, Virtual Machine Control Register

The GICV\_CTLR characteristics are:

## Purpose

Controls the behavior of virtual interrupts.

This register corresponds to the physical CPU interface register [GICC\\_CTLR](#).

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_CTLR](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_CTLR\\_EL1](#) provides equivalent functionality.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

This register is available when a GIC implementation supports interrupt virtualization.

## Attributes

GICV\_CTLR is a 32-bit register.

## Field descriptions

The GICV\_CTLR 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	EOImode	0	0	0	0	CBPR	FIQEn	AckCtl	EnableGrp1	EnableGrp0

### Bits [31:10]

Reserved, RES0.

### EOImode, bit [9]

Controls the behavior associated with the [GICV\\_EOIR](#), [GICV\\_AEOIR](#), and [GICV\\_DIR](#) registers:

EOImode	Meaning
0	Writes to <a href="#">GICV_EOIR</a> and <a href="#">GICV_AEOIR</a> perform priority drop and deactivate interrupt operations simultaneously. Behavior on a write to <a href="#">GICV_DIR</a> is UNPREDICTABLE. When it has completed processing the interrupt, the virtual machine writes to <a href="#">GICV_EOIR</a> or <a href="#">GICV_AEOIR</a> to deactivate the interrupt. The write updates the List registers and causes the virtual CPU interface to signal the interrupt completion to the physical Distributor.
1	Writes to <a href="#">GICV_EOIR</a> and <a href="#">GICV_AEOIR</a> perform priority drop operation only. Writes to <a href="#">GICV_DIR</a> perform deactivate interrupt operation only. At some point during interrupt processing, the virtual machine writes to <a href="#">GICV_EOIR</a> or <a href="#">GICV_AEOIR</a> . This write drops the priority of the virtual interrupt by updating its entry in the List registers. When it has completed processing the interrupt, the virtual machine writes to <a href="#">GICV_DIR</a> to deactivate the interrupt. The write updates the List registers and causes the virtual CPU interface to signal the interrupt completion to the Distributor.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Bits [8:5]

Reserved, RES0.

#### CBPR, bit [4]

Controls whether [GICV\\_BPR](#) affects both Group 0 and Group 1 interrupts:

CBPR	Meaning
0	<a href="#">GICV_BPR</a> affects Group 0 virtual interrupts only. <a href="#">GICV_ABPR</a> affects Group 1 virtual interrupts only.
1	<a href="#">GICV_BPR</a> affects both Group 0 and Group 1 virtual interrupts.

See Priority grouping for more information.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### FIQEn, bit [3]

FIQ Enable. Controls whether Group 0 virtual interrupts are presented as virtual FIQs:

FIQEn	Meaning
0	Group 0 virtual interrupts are presented as virtual IRQs.
1	Group 0 virtual interrupts are presented as virtual FIQs.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### AckCtl, bit [2]

ARM deprecates use of this bit. ARM strongly recommends that software is written to operate with this bit always cleared to 0.

Acknowledge control. When the highest priority interrupt is Group 1, determines whether [GICV\\_IAR](#) causes the CPU interface to acknowledge the interrupt or returns the spurious identifier 1022, and whether [GICV\\_HPPIR](#) returns the interrupt ID or the special identifier 1022.

AckCtl	Meaning
0	If the highest priority pending interrupt is Group 1, a read of <a href="#">GICV_IAR</a> or <a href="#">GICV_HPPIR</a> returns an interrupt ID of 1022.
1	If the highest priority pending interrupt is Group 1, a read of <a href="#">GICV_IAR</a> or <a href="#">GICV_HPPIR</a> returns the interrupt ID of the corresponding interrupt.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### EnableGrp1, bit [1]

Enables the signaling of Group 1 virtual interrupts by the virtual CPU interface to the virtual machine:

EnableGrp1	Meaning
0	Signaling of Group 1 interrupts is disabled.
1	Signaling of Group 1 interrupts is enabled.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### EnableGrp0, bit [0]

Enables the signaling of Group 0 virtual interrupts by the virtual CPU interface to the virtual machine:

EnableGrp0	Meaning
0	Signaling of Group 0 interrupts is disabled.
1	Signaling of Group 0 interrupts is enabled.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the GICV\_CTLR

GICV\_CTLR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual CPU interface	0x0000

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICV\_DIR, Virtual Machine Deactivate Interrupt Register

The GICV\_DIR characteristics are:

## Purpose

Deactivates a specified virtual interrupt in the [GICH\\_LR<n>](#) List registers.

This register corresponds to the physical CPU interface register [GICC\\_DIR](#).

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_DIR](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_DIR\\_EL1](#) provides equivalent functionality.

Writes to this register are valid only when [GICV\\_CTLR](#).EOImode == 1. Writes to this register are otherwise UNPREDICTABLE.

When affinity routing is enabled, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICV\_DIR is a 32-bit register.

## Field descriptions

The GICV\_DIR 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	INTID																								

### Bits [31:25]

Reserved, RES0.

### INTID, bits [24:0]

The INTID of the signaled interrupt.

---

#### Note

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

---

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

When the virtual machine writes to this register, the specified interrupt in the List registers is changed from active to inactive, or from active and pending to pending. If the specified interrupt is present in the List registers but is not in either the active or active and pending states, the effect is UNPREDICTABLE. If the specified interrupt is not present in the List registers, [GICH\\_HCR](#).EOIcount is incremented, potentially generating a maintenance interrupt.

---

#### Note

If the specified interrupt is not present in the List registers, the virtual machine cannot recover the INTID. Therefore, the hypervisor must ensure that, when [GICV\\_CTLR](#).EOImode == 1, no more than one active interrupt is transferred from the List registers into a software list. If more than one active interrupt that is not stored in the List registers exists, the hypervisor must handle accesses to GICV\_DIR in software, typically by trapping these accesses.

---

If the corresponding [GICH\\_LR<n>](#).HW == 1, indicating a hardware interrupt, then a deactivate request is sent to the physical Distributor, identifying the physical INTID from the corresponding field in the List register. This effect is identical to a Non-secure write to [GICC\\_DIR](#) from the PE having that physical INTID. This means that if the corresponding physical interrupt is marked as Group 0, the request is ignored.

---

#### Note

Interrupt deactivation using this register is based on the provided INTID, with no requirement to deactivate interrupts in any particular order. A single register is therefore used to deactivate both Group 0 and Group 1 interrupts.

---

## Accessing the GICV\_DIR

GICV\_DIR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual CPU interface	0x1000

# GICV\_EOIR, Virtual Machine End Of Interrupt Register

The GICV\_EOIR characteristics are:

## Purpose

A write to this register performs a priority drop for the specified Group 0 virtual interrupt and, if [GICV\\_CTLR.EOImode](#) == 0, also deactivates the interrupt.

This register corresponds to the physical CPU interface register [GICC\\_EOIR](#).

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_EOIR0](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_EOIR0\\_EL1](#) provides equivalent functionality.

This register is used for Group 0 interrupts only. [GICV\\_AEOIR](#) provides equivalent functionality for Group 1 interrupts.

When affinity routing is enabled, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICV\_EOIR is a 32-bit register.

## Field descriptions

The GICV\_EOIR 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	INTID																										

### Bits [31:25]

Reserved, RES0.

### INTID, bits [24:0]

The INTID of the signaled interrupt.

---

#### Note

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

---



When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

The behavior of this register depends on the setting of [GICV\\_CTLR.EOImode](#):

<a href="#">GICV_CTLR.EOImode</a>	Behavior
0	Both the priority drop and the deactivate interrupt effects occur.
1	Only the priority drop effect occurs.

A successful EOI request means that:

- The highest priority bit in [GICH\\_APR<n>](#) is cleared, causing the running priority to drop.
- If the appropriate [GICV\\_CTLR.EOImode](#) bit == 0, the interrupt is deactivated in the corresponding List register [GICH\\_LR<n>](#). If [GICH\\_LR<n>.HW](#) == 1, indicating the INTID corresponds to a hardware interrupt, a deactivate request is also sent to the physical Distributor, identifying the physical INTID from the corresponding field in the List register. This effect is identical to a Non-secure write to [GICC\\_DIR](#) from the PE having that physical INTID. This means that if the corresponding physical interrupt is marked as Group 0, and [GICD\\_CTLR.DS](#) == 0, the deactivation request is ignored. See [GICC\\_EOIR](#) for more information.

#### Note

Only Group 1 interrupts can target the hypervisor, and therefore only Group 1 interrupts are deactivated in the Distributor.

## Accessing the GICV\_EOIR

GICV\_EOIR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual CPU interface	0x0010

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICV\_HPPIR, Virtual Machine Highest Priority Pending Interrupt Register

The GICV\_HPPIR characteristics are:

## Purpose

Provides the INTID of the highest priority pending Group 0 virtual interrupt in the List registers.

This register corresponds to the physical CPU interface register [GICC\\_HPPIR](#).

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_HPPIR0](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_HPPIR0\\_EL1](#) provides equivalent functionality.

This register is used for Group 0 interrupts only. [GICV\\_AHPPIR](#) provides equivalent functionality for Group 1 interrupts.

When affinity routing is enabled, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICV\_HPPIR is a 32-bit register.

## Field descriptions

The GICV\_HPPIR 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	INTID																										

### Bits [31:25]

Reserved, RES0.

### INTID, bits [24:0]

The INTID of the signaled interrupt.

---

#### Note

---

---

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

---

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

Reads of the GICC\_HPPIR that do not return a valid INTID return a spurious INTID, 1022 or 1023. See Special INTIDs.

Highest priority pending interrupt Group	GICV_HPPIR read	<a href="#">GICV_CTLR</a> .AckCtl	Returned INTID
1	Non-secure	x	ID of Group 1 interrupt
1	Secure	0	1022
1	Secure	1	ID of Group 1 interrupt
0	Non-secure	x	1023
0	Secure	x	ID of Group 0 interrupt
No pending interrupts	x	x	1023

If the CPU interface supports only a single Security state, the entries that apply to Secure reads describe the behavior.

## Accessing the GICV\_HPPIR

GICV\_HPPIR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual CPU interface	0x0018

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICV\_IAR, Virtual Machine Interrupt Acknowledge Register

The GICV\_IAR characteristics are:

## Purpose

Provides the INTID of the signaled Group 0 virtual interrupt. A read of this register by the PE acts as an acknowledge for the interrupt.

This register corresponds to the physical CPU interface register [GICC\\_IAR](#).

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_IAR0](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_IAR0\\_EL1](#) provides equivalent functionality.

This register is used for Group 0 interrupts only. [GICV\\_AIAR](#) provides equivalent functionality for Group 1 interrupts.

When affinity routing is enabled, it is a programming error to use memory-mapped registers to access the GIC.

## Configuration

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICV\_IAR is a 32-bit register.

## Field descriptions

The GICV\_IAR 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	INTID																								

### Bits [31:25]

Reserved, RES0.

### INTID, bits [24:0]

The INTID of the signaled interrupt.

---

#### Note

INTIDs 1020-1023 are reserved and convey additional information such as spurious interrupts.

---

When affinity routing is not enabled:

- Bits [23:13] are RES0.
- For SGIs, bits [12:10] identify the CPU interface corresponding to the source PE. For all other interrupts these bits are RES0.

When the virtual machine writes to this register, the virtual CPU interface acknowledges the highest priority pending virtual interrupt and sets the state in the corresponding List register to active. The appropriate bit in the active priorities register [GICH\\_APR<n>](#) is set to 1.

If [GICH\\_LR<n>](#).HW == 0, indicating that the interrupt is software-triggered, then bits [12:10] of [GICH\\_LR<n>](#) are returned in bits [12:10] of GICV\_IAR. Otherwise bits [12:10] are RES0.

A read of this register returns the spurious INTID 1023 if either of the following is true:

- There are no pending interrupts of sufficiently high priority value to be signaled to the PE with the virtual CPU interface enabled and [GICH\\_HCR](#).En == 1.
- Interrupt signaling by the virtual CPU interface is disabled.

A read of this register returns the spurious INTID 1022 if the highest priority pending interrupt is Group 1 and [GICV\\_CTLR](#).AckCtl == 0.

## Accessing the GICV\_IAR

GICV\_IAR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual CPU interface	0x000C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICV\_IIDR, Virtual Machine CPU Interface Identification Register

The GICV\_IIDR characteristics are:

## Purpose

Provides information about the implementer and revision of the virtual CPU interface.

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

## Configuration

This register is available in all configurations of the GIC. If the GIC implementation supports two Security states this register is Common.

## Attributes

GICV\_IIDR is a 32-bit register.

## Field descriptions

The GICV\_IIDR 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
ProductID												Architecture_version			Revision			Implementer													

### ProductID, bits [31:20]

An IMPLEMENTATION DEFINED product identifier.

### Architecture\_version, bits [19:16]

The version of the GIC architecture that is implemented.

Architecture_version	Meaning
0001	GICv1.
0010	GICv2.
0011	GICv3 memory-mapped interface supported. Support for the System register interface is discoverable from PE registers ID_PFR1 and ID_AA64PFR0_EL1.
0100	GICv4 memory-mapped interface supported. Support for the System register interface is discoverable from PE registers ID_PFR1 and ID_AA64PFR0_EL1.

Other values are reserved.

### Revision, bits [15:12]

An IMPLEMENTATION DEFINED revision number for the CPU interface.

**Implementer, bits [11:0]**

Contains the JEP106 code of the company that implemented the CPU interface.

- Bits [11:8] are the JEP106 continuation code of the implementer. For an ARM implementation, this field is 0x4.
- Bit [7] is always 0.
- Bits [6:0] are the JEP106 identity code of the implementer. For an ARM implementation, bits [7:0] are therefore 0x3B.

## Accessing the GICV\_IIDR

GICV\_IIDR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual CPU interface	0x00FC

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICV\_PMR, Virtual Machine Priority Mask Register

The GICV\_PMR characteristics are:

## Purpose

This register provides a virtual interrupt priority filter. Only virtual interrupts with a higher priority than the value in this register are signaled to the PE.

### Note

Higher interrupt priority corresponds to a lower value of the Priority field.

This register corresponds to the physical CPU interface register [GICC\\_PMR](#).

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_PMR](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_PMR\\_EL1](#) provides equivalent functionality.

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

This register is available when the GIC implementation supports interrupt virtualization.

The Priority field of this register is aliased to [GICH\\_VMCR.VMPR](#), to enable state to be switched easily between virtual machines during context-switching.

## Attributes

GICV\_PMR is a 32-bit register.

## Field descriptions

The GICV\_PMR 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	Priority									

### Bits [31:8]

Reserved, RES0.



**Priority, bits [7:0]**

The priority mask level for the virtual CPU interface. If the priority of the interrupt is higher than the value indicated by this field, the interface signals the interrupt to the PE.

If the GIC implementation supports fewer than 256 priority levels some bits might be RAZ/WI, as follows:

- For 128 supported levels, bit [0] = 0b0.
- For 64 supported levels, bits [1:0] = 0b00.
- For 32 supported levels, bits [2:0] = 0b000.
- For 16 supported levels, bits [3:0] = 0b0000.

See Interrupt prioritization, section 4.8 of the GICv3 Architecture Specification for more information.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Accessing the GICV\_PMR**

GICV\_PMR can be accessed through its memory-mapped interface:

Component	Offset
GIC Virtual CPU interface	0x0004

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICV\_RPR, Virtual Machine Running Priority Register

The GICV\_RPR characteristics are:

## Purpose

This register indicates the running priority of the virtual CPU interface.

This register corresponds to the physical CPU interface register [GICC\\_RPR](#).

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

This register is used only when System register access is not enabled. When System register access is enabled:

- For AArch32 implementations, [ICC\\_RPR](#) provides equivalent functionality.
- For AArch64 implementations, [ICC\\_RPR\\_EL1](#) provides equivalent functionality.

Depending on the implementation, if no bits are set to 1 in [GICH\\_APR<n>](#), indicating no active virtual interrupts in the virtual CPU interface, the priority reads as 0xFF or 0xF8 to reflect the number of supported interrupt priority bits defined by [GICH\\_VTR.PRIBits](#).

## Configuration

This register is available when the GIC implementation supports interrupt virtualization.

## Attributes

GICV\_RPR is a 32-bit register.

## Field descriptions

The GICV\_RPR 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	Priority									

### Bits [31:8]

Reserved, RES0.

### Priority, bits [7:0]

The current running priority on the virtual CPU interface.

## Accessing the GICV\_RPR

GICV\_RPR can be accessed through its memory-mapped interface:

Component	Offset
-----------	--------

GIC Virtual CPU interface	0x0014
------------------------------	--------

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GICV\_STATUSR, Virtual Machine Error Reporting Status Register

The GICV\_STATUSR characteristics are:

## Purpose

Provides software with a mechanism to detect:

- Accesses to reserved locations.
- Writes to read-only locations.
- Reads of write-only locations.

This register is part of the GIC virtual CPU interface registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

This is an optional register. If the register is implemented, [GICC\\_STATUSR](#) must also be implemented. If the register is not implemented, the location is RAZ/WI.

This register is used only when System register access is not enabled. If System register access is enabled, this register is not updated. Equivalent function might be provided by appropriate traps and exceptions.

## Configuration

In systems where this register is implemented, ARM expects that when a virtual machine is scheduled, the hypervisor ensures that this register is cleared to 0. The hypervisor might check for illegal accesses when the virtual machine is unscheduled.

## Attributes

GICV\_STATUSR is a 32-bit register.

## Field descriptions

The GICV\_STATUSR 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	WROD	RWOD	WRD	RRD

### Bits [31:4]

Reserved, RES0.

### WROD, bit [3]

Write to an RO location.

WROD	Meaning
0	Normal operation.
1	A write to an RO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**RWOD, bit [2]**

Read of a WO location.

<b>RWOD</b>	<b>Meaning</b>
0	Normal operation.
1	A read of a WO location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**WRD, bit [1]**

Write to a reserved location.

<b>WRD</b>	<b>Meaning</b>
0	Normal operation.
1	A write to a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**RRD, bit [0]**

Read of a reserved location.

<b>RRD</b>	<b>Meaning</b>
0	Normal operation.
1	A read of a reserved location has been detected.

When a violation is detected, software must write 1 to this register to reset it.

**Accessing the GICV\_STATUSR**

GICV\_STATUSR can be accessed through its memory-mapped interface:

<b>Component</b>	<b>Offset</b>
GIC Virtual CPU interface	0x002C

# GITS\_BASER<n>, ITS Translation Table Descriptors, n = 0 - 7

The GITS\_BASER<n> characteristics are:

## Purpose

Specifies the base address and size of the ITS translation tables.

This register is part of the GIC ITS registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

## Configuration

Some or all RW fields of this register have defined reset values.

A copy of this register is provided for each ITS translation table.

Bits [63:32] and bits [31:0] are accessible independently.

A maximum of 8 GITS\_BASER<n> registers can be provided. Unimplemented registers are RES0.

When [GITS\\_CTLR.Enabled](#) == 1 or [GITS\\_CTLR.Quiescent](#) == 0, writing this register is UNPREDICTABLE.

## Attributes

GITS\_BASER<n> is a 64-bit register.

## Field descriptions

The GITS\_BASER<n> 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
Valid	Indirect	InnerCache	Type	OuterCache	Entry_Size	Physical_Address																									
Physical_Address																Shareability	Page_Size	Size													
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Valid, bit [63]

Indicates whether software has allocated memory for the translation table:

Valid	Meaning
0	No memory is allocated for the translation table. The ITS discards any writes to the interrupt translation page when either: <ul style="list-style-type: none"> <li>GITS_BASER&lt;n&gt;.Type specifies any valid table entry type other than interrupt collections, that is, any value other than 100.</li> <li>GITS_BASER&lt;n&gt;.Type specifies an interrupt collection and <a href="#">GITS_TYPER.HCC</a> == 0.</li> </ul>
1	Memory is allocated to the translation table.

When this register has an architecturally-defined reset value, this field resets to 0.

**Indirect, bit [62]**

This field indicates whether an implemented register specifies a single, flat table or a two-level table where the first level contains a list of descriptors.

**Note**

This field is RAZ/WI for implementations that only support flat tables.

Indirect	Meaning
0	Single Level. The Size field indicates the number of pages used by the ITS to store data associated with each table entry.
1	Two Level. The Size field indicates the number of pages which contain an array of 64-bit descriptors to pages that are used to store the data associated with each table entry. A little endian memory order model is used.

See The ITS tables for more information.

This field is RAZ/WI for GIC implementations that only support flat tables.

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.

**InnerCache, bits [61:59]**

Indicates the Inner Cacheability attributes of accesses to the table. The possible values of this field are:

InnerCache	Meaning
000	Device-nGnRnE.
001	Normal Inner Non-cacheable.
010	Normal Inner Cacheable Read-allocate, Write-through.
011	Normal Inner Cacheable Read-allocate, Write-back.
100	Normal Inner Cacheable Write-allocate, Write-through.
101	Normal Inner Cacheable Write-allocate, Write-back.
110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Type, bits [58:56]**

Read only. Specifies the type of entity that requires entries in the corresponding translation table. The possible values of the field are:

Type	Meaning
000	Unimplemented. This register does not correspond to a translation table.
001	Devices. This register corresponds to a translation table that scales with the width of the DeviceID. Only a single GITS_BASER<n> register reports this type.
010	vPEs. GICv4 only. This register corresponds to a translation table that scales with the number of vPEs in the system. The translation table requires (ENTRY_SIZE * N) bytes of memory, where N is the number of vPEs in the system. Only a single GITS_BASER<n> register reports this type.
100	Interrupt collections. This register corresponds to a translation table that scales with the number of interrupt collections in the system. The translation table requires (ENTRY_SIZE * N) bytes of memory, where N is the number of interrupt collections. Not more than one GITS_BASER<n> register will report this type.

Other values are reserved.

**Note**

The minimum number of entries that an ITS must support is N+1, where N is the number of physical PEs in the system.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**OuterCache, bits [55:53]**

Indicates the Outer Cacheability attributes of accesses to the table. The possible values of this field are:

OuterCache	Meaning
000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
001	Normal Outer Non-cacheable.
010	Normal Outer Cacheable Read-allocate, Write-through.
011	Normal Outer Cacheable Read-allocate, Write-back.
100	Normal Outer Cacheable Write-allocate, Write-through.
101	Normal Outer Cacheable Write-allocate, Write-back.
110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

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.

**Entry\_Size, bits [52:48]**

Read-only. Specifies the number of bytes per translation table entry, minus one.

**Physical\_Address, bits [47:12]**

Physical Address. When Page\_Size is 4KB or 16KB:

- Bits [51:48] of the base physical address are zero.
- This field provides bits[47:12] of the base physical address of the table.
- Bits[11:0] of the base physical address are zero.
- The address must be aligned to the size specified in the Page Size field. Otherwise the effect is CONSTRAINED UNPREDICTABLE, and can be one of the following:
  - Bits[X:12], where X is derived from the page size, are treated as zero.
  - The value of bits[X:12] are used when calculating the address of a table access.

When Page\_Size is 64KB:

- Bits[47:16] of the register provide bits[47:16] of the base physical address of the table.
- Bits[15:12] of the register provide bits[51:48] of the base physical address of the table.
- Bits[15:0] of the base physical address are 0.

In implementations that support fewer than 52 bits of physical address, any unimplemented upper bits might be RAZ/WI.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Shareability, bits [11:10]**

Indicates the Shareability attributes of accesses to the table. The possible values of this field are:

Shareability	Meaning
00	Non-shareable.
01	Inner Shareable.
10	Outer Shareable.
11	Reserved. Treated as 00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

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.



**Page\_Size, bits [9:8]**

The size of page that the translation table uses:

Page_Size	Meaning
00	4KB.
01	16KB.
10	64KB.
11	Reserved. Treated as 10.

**Note**

If the GIC implementation supports only a single, fixed page size, this field might be RO.

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.

**Size, bits [7:0]**

The number of pages of physical memory allocated to the table, minus one. GITS\_BASER<n>.Page\_Size specifies the size of each page.

If GITS\_BASER<n>.Type == 0, this field is RAZ/WI.

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.

**Accessing the GITS\_BASER<n>**

GITS\_BASER<n> can be accessed through its memory-mapped interface:

Component	Offset
GIC ITS control	$0 \times 0100 + 8n$

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GITS\_CBASER, ITS Command Queue Descriptor

The GITS\_CBASER characteristics are:

## Purpose

Specifies the base address and size of the ITS command queue.

This register is part of the GIC ITS registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

When [GITS\\_CTLR.Enabled](#) == 1 or [GITS\\_CTLR.Quiescent](#) == 0, writing this register is UNPREDICTABLE.

## Configuration

Some or all RW fields of this register have defined reset values.

Bits [63:32] and bits [31:0] are accessible separately.

## Attributes

GITS\_CBASER is a 64-bit register.

## Field descriptions

The GITS\_CBASER 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
Valid	0	InnerCache			0	0	0	OuterCache			0	Physical_Address																			
Physical_Address																				Shareability		0	0	Size							
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Valid, bit [63]

Indicates whether software has allocated memory for the command queue:

Valid	Meaning
0	No memory is allocated for the command queue.
1	Memory is allocated to the command queue.

When this register has an architecturally-defined reset value, this field resets to 0.

### Bit [62]

Reserved, RES0.

### InnerCache, bits [61:59]

Indicates the Inner Cacheability attributes of accesses to the command queue. The possible values of this field are:

InnerCache	Meaning
000	Device-nGnRnE.
001	Normal Inner Non-cacheable.
010	Normal Inner Cacheable Read-allocate, Write-through.
011	Normal Inner Cacheable Read-allocate, Write-back.
100	Normal Inner Cacheable Write-allocate, Write-through.
101	Normal Inner Cacheable Write-allocate, Write-back.
110	Normal Inner Cacheable Read-allocate, Write-allocate, Write-through.
111	Normal Inner Cacheable Read-allocate, Write-allocate, Write-back.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Bits [58:56]

Reserved, RES0.

#### OuterCache, bits [55:53]

Indicates the Outer Cacheability attributes of accesses to the command queue. The possible values of this field are:

OuterCache	Meaning
000	Memory type defined in InnerCache field. For Normal memory, Outer Cacheability is the same as Inner Cacheability.
001	Normal Outer Non-cacheable.
010	Normal Outer Cacheable Read-allocate, Write-through.
011	Normal Outer Cacheable Read-allocate, Write-back.
100	Normal Outer Cacheable Write-allocate, Write-through.
101	Normal Outer Cacheable Write-allocate, Write-back.
110	Normal Outer Cacheable Read-allocate, Write-allocate, Write-through.
111	Normal Outer Cacheable Read-allocate, Write-allocate, Write-back.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

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 [52]

Reserved, RES0.

#### Physical\_Address, bits [51:12]

Bits [51:12] of the base physical address of the command queue. Bits [11:0] of the base address are 0.

In implementations supporting fewer than 52 bits of physical address, unimplemented upper bits are RES0.

If bits [15:12] are not all zeros, behavior is a CONSTRAINED UNPREDICTABLE choice:

- Bits [15:12] are treated as if all the bits are zero. The value read back from those bits is either the value written or zero.
- The result of the calculation of an address for a command queue read can be corrupted.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### Shareability, bits [11:10]

Indicates the Shareability attributes of accesses to the command queue. The possible values of this field are:

Shareability	Meaning
00	Non-shareable.
01	Inner Shareable.
10	Outer Shareable.
11	Reserved. Treated as 00.

It is IMPLEMENTATION DEFINED whether this field has a fixed value or can be programmed by software. Implementing this field with a fixed value is deprecated.

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 [9:8]

Reserved, RES0.

#### Size, bits [7:0]

The number of 4KB pages of physical memory allocated to the command queue, minus one.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

The command queue is a circular buffer and wraps at Physical Address [47:0] + (4096 \* (Size + 1)).

---

#### Note

When this register is successfully written, the value of [GITS\\_CREADR](#) is set to zero.

---

## Accessing the GITS\_CBASER

GITS\_CBASER can be accessed through its memory-mapped interface:

Component	Offset
GIC ITS control	0x0080 - 0x0084

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GITS\_CREADR, ITS Read Register

The GITS\_CREADR characteristics are:

## Purpose

Specifies the offset from [GITS\\_CBASER](#) where the ITS reads the next ITS command.

This register is part of the GIC ITS registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

## Configuration

This register is cleared to 0 when a value is written to [GITS\\_CBASER](#).

Bits [63:32] and bits [31:0] are accessible separately.

## Attributes

GITS\_CREADR is a 64-bit register.

## Field descriptions

The GITS\_CREADR 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
0	0	0	0	0	0	0	0	0	0	0	0	Offset													0	0	0	0	Stalled		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:20]

Reserved, RES0.

### Offset, bits [19:5]

Bits [19:5] of the offset from [GITS\\_CBASER](#). Bits [4:0] of the offset are zero.

### Bits [4:1]

Reserved, RES0.

### Stalled, bit [0]

Reports whether the processing of commands is stalled because of a command error.

Stalled	Meaning
0	ITS command queue is not stalled because of a command error.
1	ITS command queue is stalled because of a command error.

See The ITS Command Interface for more information.

## Accessing the GITS\_CREADR

GITS\_CREADR can be accessed through its memory-mapped interface:

Component	Offset
GIC ITS control	0x0090 - 0x0094

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GITS\_CTLR, ITS Control Register

The GITS\_CTLR characteristics are:

## Purpose

Controls the operation of an ITS.

This register is part of the GIC ITS registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

## Configuration

Some or all RW fields of this register have defined reset values.

The ITS\_Number (bits [7:4]) and bit [1] fields apply only in GICv4 implementations, and are RES0 in GICv3 implementations.

## Attributes

GITS\_CTLR is a 32-bit register.

## Field descriptions

The GITS\_CTLR 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
Quiescent	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ITS_Number	0	0	0	0	ImDe	Enabled	

### Quiescent, bit [31]

Read-only. Indicates completion of all ITS operations when GITS\_CTLR.Enabled == 0.

Quiescent	Meaning
0	The ITS is not quiescent and cannot be powered down.
1	The ITS is quiescent and can be powered down.

For the ITS to be quiescent, there must be no transactions in progress. In addition, all operations required to ensure that mapping data is consistent with external memory must be complete.

### Note

In distributed GIC implementations, this bit is set to 1 only after the ITS forwards any operations that have not yet been completed to the Redistributors and receives confirmation that all such operations have reached the appropriate Redistributor.

When GITS\_CTLR.Enabled==1 the value of GITS\_CTLR.Quiescent is UNKNOWN.

When this register has an architecturally-defined reset value, this field resets to 1.

**Bits [30:8]**

Reserved, RES0.

**ITS\_Number, bits [7:4]**

In GICv3 implementations this field is RES0.

In GICv4 implementations with more than one ITS instance, this field indicates the ITS number for use with VMOVP.

It is IMPLEMENTATION DEFINED whether this field is programmable or RO.

If this field is programmable, changing this field when GITS\_CTLR.Quiescent==0 or GITS\_CTLR.Enabled==1 is UNPREDICTABLE.

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 [3:2]**

Reserved, RES0.

**ImDe, bit [1]**

In GICv3 implementations this bit is RES0.

In GICv4 implementations this bit is IMPLEMENTATION DEFINED.

When this register has an architecturally-defined reset value, if this field is implemented as an RW field, it resets to 0.

**Enabled, bit [0]**

Controls whether the ITS is enabled:

Enabled	Meaning
0	The ITS is not enabled. Writes to <a href="#">GITS_TRANSLATER</a> are ignored and no further command queue entries are processed.
1	The ITS is enabled. Writes to <a href="#">GITS_TRANSLATER</a> result in interrupt translations and the command queue is processed.

If a write to this register changes this field from 1 to 0, the ITS must ensure that both:

- Any caches containing mapping data are made consistent with external memory.
- GITS\_CTLR.Quiescent == 0 until all caches are consistent with external memory.

When this register has an architecturally-defined reset value, this field resets to 0.

**Accessing the GITS\_CTLR**

GITS\_CTLR can be accessed through its memory-mapped interface:

Component	Offset
GIC ITS control	0x0000



# GITS\_CWRITER, ITS Write Register

The GITS\_CWRITER characteristics are:

## Purpose

Specifies the offset from [GITS\\_CBASER](#) where software writes the next ITS command.

This register is part of the GIC ITS registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RW	RW	RW

## Configuration

RW fields in this register reset to architecturally UNKNOWN values.

Bits [63:32] and bits [31:0] are accessible separately.

## Attributes

GITS\_CWRITER is a 64-bit register.

## Field descriptions

The GITS\_CWRITER 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
0	0	0	0	0	0	0	0	0	0	0	0	Offset														0	0	0	0	Retry	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:20]

Reserved, RES0.

### Offset, bits [19:5]

Bits [19:5] of the offset from [GITS\\_CBASER](#). Bits [4:0] of the offset are zero.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### Bits [4:1]

Reserved, RES0.

### Retry, bit [0]

Writing this bit has the following effects:

Retry	Meaning
0	No effect on the processing commands by the ITS.
1	Restarts the processing of commands by the ITS if it stalled because of a command error.
<b>Note</b> If the processing of commands is not stalled because of a command error, writing 1 to this bit has no effect.	

When read, this bit is RES0.

See The ITS Command Interface for more information.

If GITS\_CWRITER is written with a value outside of the valid range specified by [GITS\\_CBASER.Physical\\_Address](#) and [GITS\\_CBASER.Size](#), behavior is a CONSTRAINED UNPREDICTABLE choice, as follows:

- The command queue is considered invalid, and no further commands are processed until GITS\_CWRITER is written with a value that is in the valid range.
- The value is treated as a valid UNKNOWN value.

An implementation might choose to report a system error in an IMPLEMENTATION DEFINED manner.

## Accessing the GITS\_CWRITER

GITS\_CWRITER can be accessed through its memory-mapped interface:

Component	Offset
GIC ITS control	0x0088 - 0x008C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GITS\_IIDR, ITS Identification Register

The GITS\_IIDR characteristics are:

## Purpose

Provides information about the implementer and revision of the ITS.

This register is part of the GIC ITS registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

## Configuration

This register is available in all configurations of the GIC. If the GIC implementation supports two Security states, this register is Common.

## Attributes

GITS\_IIDR is a 32-bit register.

## Field descriptions

The GITS\_IIDR 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
ProductID								0	0	0	0	Variant			Revision			Implementer													

### ProductID, bits [31:24]

An IMPLEMENTATION DEFINED product identifier.

### Bits [23:20]

Reserved, RES0.

### Variant, bits [19:16]

An IMPLEMENTATION DEFINED variant number. Typically, this field is used to distinguish product variants, or major revisions of a product.

### Revision, bits [15:12]

An IMPLEMENTATION DEFINED revision number. Typically, this field is used to distinguish minor revisions of a product.

### Implementer, bits [11:0]

Contains the JEP106 code of the company that implemented the ITS:

- Bits [11:8] are the JEP106 continuation code of the implementer. For an ARM implementation, this field is 0x4.

- Bit [7] is always 0.
- Bits [6:0] are the JEP106 identity code of the implementer. For an ARM implementation, bits [7:0] are therefore 0x3B.

## Accessing the GITS\_IIDR

GITS\_IIDR can be accessed through its memory-mapped interface:

Component	Offset
GIC ITS control	0x0004

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GITS\_TRANSLATER, ITS Translation Register

The GITS\_TRANSLATER characteristics are:

## Purpose

Written by a requesting a Device to signal an interrupt for translation by the ITS.

This register is part of the GIC ITS registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
WO	WO	WO

16-bit access to bits [15:0] of this register must be supported. When this register is written by a 16-bit transaction, bits [31:16] are written as zero.

Implementations must ensure that:

- A unique DeviceID is provided for each requesting device, and the DeviceID is presented to the ITS when a write to this register occurs in a manner that cannot be spoofed by any agent capable of performing writes.
- The DeviceID presented corresponds to the DeviceID field in the ITS commands.

Writes to this register are ignored if any of the following are true:

- [GITS\\_CTLR](#).Enabled == 0.
- The presented DeviceID is not mapped to an Interrupt Translation Table.
- The DeviceID is larger than the supported size.
- The DeviceID is mapped to an Interrupt Translation Table, but the EventID is outside the range specified by MAPD.
- The EventID is mapped to an Interrupt Translation Table and the EventID is within the range specified by MAPD, but the EventID is unmapped.

Translation requests that result from writes to this register are subject to certain ordering rules. See Ordering of translations following writes to GITS\_TRANSLATER for more information.

## Configuration

This register is at the same offset as [GICD\\_SETSPI\\_NSR](#) in the Distributor, and is at the same offset as [GICR\\_SETLPIR](#) in the Redistributor.

## Attributes

GITS\_TRANSLATER is a 32-bit register.

## Field descriptions

The GITS\_TRANSLATER 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
																<a href="#">EventID</a>															

### EventID, bits [31:0]

An identifier corresponding to the interrupt to be translated.

---

**Note**

The size of the EventID is DeviceID specific, and set when the DeviceID is mapped to an ITT (using MAPD).

---

The number of EventID bits implemented is reported by [GITS\\_TYPER.ID\\_bits](#). If a write specifies non-zero identifiers bits outside this range behavior is a CONSTRAINED UNPREDICTABLE choice between:

- Non-zero identifier bits outside the supported range are ignored.
- The write is ignored.

The DeviceID presented to an ITS is used to index a device table. The device table maps the DeviceID to an interrupt translation table for that device.

## Accessing the GITS\_TRANSLATER

GITS\_TRANSLATER can be accessed through its memory-mapped interface:

Component	Offset
GIC ITS translation	0x0040

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# GITS\_TYPER, ITS Type Register

The GITS\_TYPER characteristics are:

## Purpose

Specifies the features that an ITS supports.

This register is part of the GIC ITS registers functional group.

## Usage constraints

This register is accessible as follows:

Security disabled	Secure	Non-secure
RO	RO	RO

## Configuration

There are no configuration notes.

## Attributes

GITS\_TYPER is a 64-bit register.

## Field descriptions

The GITS\_TYPER 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	VMOVP	CIL	CIDbits			
HCC				0	0	0	0	PT	ASE	IS	Devbits			ID_bits			ITT_entry_size			IMPLEMENTATION DEFINED			CCT	Virtual	Physical						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### Bits [63:38]

Reserved, RES0.

### VMOVP, bit [37]

Indicates the form of the VMOVP command.

VMOVP	Meaning
0	When moving a vPE, software must issue a VMOVP on all ITSs that have mappings for that vPE. The ITSList and Sequence Number fields in the VMOVP command must ensure synchronization, otherwise behavior is UNPREDICTABLE.
1	When moving a vPE, software must only issue a VMOVP on one of the ITSs that has a mapping for that vPE. The ITSList and Sequence Number fields in the VMOVP command are RES0.

### CIL, bit [36]

Collection ID Limit.

CIL	Meaning
0	ITS supports 16-bit Collection ID, <a href="#">GITS_TYPER.CIDbits</a> is RES0.
1	<a href="#">GITS_TYPER.CIDbits</a> indicates supported Collection ID size

In implementations that do not support Collections in external memory, this bit is RES0 and the number of Collections supported is reported by [GITS\\_TYPER.HCC](#).

#### CIDbits, bits [35:32]

Number of Collection ID bits.

- The number of bits of Collection ID - 1.
- When [GITS\\_TYPER.CIL](#)==0, this field is RES0.

#### HCC, bits [31:24]

Hardware Collection Count. The number of interrupt collections supported by the ITS without provisioning of external memory.

##### Note

Collections held in hardware are unmapped at reset.

#### Bits [23:20]

Reserved, RES0.

#### PTA, bit [19]

Physical Target Addresses. Indicates the format of the target address:

PTA	Meaning
0	The target address corresponds to the PE number specified by <a href="#">GICR_TYPER.Processor_Number</a> .
1	The target address corresponds to the base physical address of the required Redistributor.

See Target addresses for more information.

#### SEIS, bit [18]

SEI support. Indicates whether the virtual CPU interface supports generation of SEIs:

SEIS	Meaning
0	The ITS does not support local generation of SEIs.
1	The ITS supports local generation of SEIs.

#### Devbits, bits [17:13]

The number of DeviceID bits implemented, minus one.

#### ID\_bits, bits [12:8]

The number of EventID bits implemented, minus one.

#### ITT\_entry\_size, bits [7:4]

Read-only. Indicates the number of bytes per translation table entry, minus one.

See the ITS command 'MAPD' for more information.



**IMPLEMENTATION DEFINED, bit [3]**

IMPLEMENTATION DEFINED.

**CCT, bit [2]**

Cumulative Collection Tables.

CCT	Meaning
0	The total number of supported collections is determined by the number of collections held in memory only.
1	The total number of supported collections is determined by number of collections that are held in memory and the number indicated by GITS_TYPER.HCC.

If GITS\_TYPER.HCC==0, or if memory backed collections are not supported (all [GITS\\_BASER<n>.Type](#) != 100), this bit is RES0.

**Virtual, bit [1]**

Indicates whether the ITS supports virtual LPIs and direct injection of virtual LPIs:

Virtual	Meaning
0	The ITS does not support virtual LPIs or direct injection of virtual LPIs.
1	The ITS supports virtual LPIs and direct injection of virtual LPIs.

This field is RES0 in GICv3 implementations.

**Physical, bit [0]**

Indicates whether the ITS supports physical LPIs:

Physical	Meaning
0	The ITS does not support physical LPIs.
1	The ITS supports physical LPIs.

This field is RES1, indicating that the ITS supports physical LPIs.

**Accessing the GITS\_TYPER**

GITS\_TYPER can be accessed through its memory-mapped interface:

Component	Offset
GIC ITS control	0x0008 - 0x000C

# 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 'Identification registers, functional group' in the ARMv8 ARM, section G4.18.1.

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

# OSLAR\_EL1, OS Lock Access Register

The OSLAR\_EL1 characteristics are:

## Purpose

Used to lock or unlock the OS lock.

This register is part of the Debug registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	EDAD	SLK	Default
Error	Error	IMP DEF	WI	WO

## Configuration

External register OSLAR\_EL1 is architecturally mapped to AArch64 System register [OSLAR\\_EL1](#).

External register OSLAR\_EL1 is architecturally mapped to AArch32 System register [DBGOSLAR](#).

OSLAR\_EL1 is in the Core power domain.

From ARMv8.2, external debug accesses to OSLAR\_EL1 are ignored and return an error when any of:

- ExternalInvasiveDebugEnabled() == FALSE.
- ExternalSecureInvasiveDebugEnabled() == FALSE and any of:
  - EL3 is not implemented and the PE is in Secure state.
  - EL3 is implemented and is using AArch64 and [MDCR\\_EL3.EDAD](#) == 1
  - EL3 is implemented and is using AArch32 and [SDCR.EDAD](#) == 1.

## Attributes

OSLAR\_EL1 is a 32-bit register.

## Field descriptions

The OSLAR\_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	0	0	<a href="#">OSLK</a>

### Bits [31:1]

Reserved, RES0.

### OSLK, bit [0]

On writes to OSLAR\_EL1, bit[0] is copied to the OS lock.

Use [EDPRSR.OSLK](#) to check the current status of the lock.

## Accessing the OSLAR\_EL1

OSLAR\_EL1 can be accessed through the external debug interface:

Component	Offset
Debug	0x300

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMAUTHSTATUS, Performance Monitors Authentication Status register

The PMAUTHSTATUS characteristics are:

## Purpose

Provides information about the state of the IMPLEMENTATION DEFINED authentication interface for Performance Monitors.

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMAUTHSTATUS is in the Debug power domain.

This register is OPTIONAL, and is required for CoreSight compliance. ARM recommends that this register is implemented.

## Attributes

PMAUTHSTATUS is a 32-bit register.

## Field descriptions

The PMAUTHSTATUS 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	SNID	SID	NSNID	NSID				

### Bits [31:8]

Reserved, RES0.

### SNID, bits [7:6]

Holds the same value as [DBGAUTHSTATUS\\_EL1](#).SNID.

### SID, bits [5:4]

Secure invasive debug. Possible values of this field are:

SID	Meaning
00	Not implemented.

All other values are reserved.

**NSNID, bits [3:2]**

Holds the same value as [DBGAUTHSTATUS\\_EL1.NSNID](#).

**NSID, bits [1:0]**

Non-secure invasive debug. Possible values of this field are:

NSID	Meaning
00	Not implemented.

All other values are reserved.

**Accessing the PMAUTHSTATUS**

PMAUTHSTATUS can be accessed through the external debug interface:

Component	Offset
PMU	0xFB8

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMCCFILTR\_EL0, Performance Monitors Cycle Counter Filter Register

The PMCCFILTR\_EL0 characteristics are:

## Purpose

Determines the modes in which the Cycle Counter, [PMCCNTR\\_EL0](#), increments.

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	RW

## Configuration

External register PMCCFILTR\_EL0 is architecturally mapped to AArch64 System register [PMCCFILTR\\_EL0](#).

External register PMCCFILTR\_EL0 is architecturally mapped to AArch32 System register [PMCCFILTR](#).

PMCCFILTR\_EL0 is in the Core power domain.

On a Warm or Cold reset RW fields in this register reset:

- To architecturally UNKNOWN values if the reset is to an Exception level that is using AArch64.
- To 0 if the reset is to an Exception level that is using AArch32.

The register is not affected by an External debug reset.

## Attributes

PMCCFILTR\_EL0 is a 32-bit register.

## Field descriptions

The PMCCFILTR\_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
P	U	NSK	NSU	NSH	M	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	

### P, bit [31]

Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are:

P	Meaning
0	Count cycles in EL1.
1	Do not count cycles in EL1.



**U, bit [30]**

User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are:

U	Meaning
0	Count cycles in EL0.
1	Do not count cycles in EL0.

**NSK, bit [29]**

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of P, cycles in Non-secure EL1 are counted.

Otherwise, cycles in Non-secure EL1 are not counted.

**NSU, bit [28]**

Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of U, cycles in Non-secure EL0 are counted.

Otherwise, cycles in Non-secure EL0 are not counted.

**NSH, bit [27]**

Non-secure EL2 (Hypervisor) filtering bit. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0.

NSH	Meaning
0	Do not count cycles in EL2.
1	Count cycles in EL2.

**M, bit [26]**

Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted.

Otherwise, cycles in Secure EL3 are not counted.

Most applications can ignore this field and set its value to 0.

**Note**

This field is not visible in the AArch32 PMCCFILTR System register.

**Bits [25:0]**

Reserved, RES0.

**Accessing the PMCCFILTR\_EL0**

PMCCFILTR\_EL0 can be accessed through the external debug interface:

Component	Offset
PMU	0x47C



# PMCCNTR\_EL0, Performance Monitors Cycle Counter

The PMCCNTR\_EL0 characteristics are:

## Purpose

Holds the value of the processor Cycle Counter, CCNT, that counts processor clock cycles. See 'Time as measured by the Performance Monitors cycle counter' in the ARMv8 ARM, section D5 for more information.

[PMCCFILTR\\_EL0](#) determines the modes and states in which the PMCCNTR\_EL0 can increment.

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EPMA	SLK	Default
Error	Error	Error	Error	RO	RW

## Configuration

External register PMCCNTR\_EL0 is architecturally mapped to AArch64 System register [PMCCNTR\\_EL0](#).

External register PMCCNTR\_EL0 is architecturally mapped to AArch32 System register [PMCCNTR](#).

PMCCNTR\_EL0 is in the Core power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply on a Warm or Cold reset. The register is not affected by an External debug reset.

## Attributes

PMCCNTR\_EL0 is a 64-bit register.

## Field descriptions

The PMCCNTR\_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
																CCNT															
																CCNT															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### CCNT, bits [63:0]

Cycle count. Depending on the values of [PMCR\\_EL0](#).{LC,D}, the cycle count increments in one of the following ways:

- Every processor clock cycle.
- Every 64th processor clock cycle.

Writing 1 to [PMCR\\_EL0](#).C sets this field to 0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the PMCCNTR\_EL0

PMCCNTR\_EL0[31:0] can be accessed through the external debug interface:

Component	Offset
PMU	0x0F8

PMCCNTR\_EL0[63:32] can be accessed through the external debug interface:

Component	Offset
PMU	0x0FC

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMCEID0, Performance Monitors Common Event Identification register 0

The PMCEID0 characteristics are:

## Purpose

Defines which common architectural and common microarchitectural feature events in the range 0x000 to 0x01F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

**Note**

This view of the register has previously been 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:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<a href="#">ID[31:0]</a>																															

**ID[31:0], bits [31:0]**

PMCEID0[n] maps to event n. For a list of event numbers and descriptions, see 'Event numbers and mnemonics' in the ARM ARM, section D5.10.

For each bit:

ID[31:0]	Meaning
0	The common event is not implemented.
1	The common event is implemented.

Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture.

Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.

## Accessing the PMCEID0

PMCEID0 can be accessed through the external debug interface:

Component	Offset
PMU	0xE20

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMCEID1, Performance Monitors Common Event Identification register 1

The PMCEID1 characteristics are:

## Purpose

Defines which common architectural and common microarchitectural feature events in the range 0x020 to 0x03F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

### Note

This view of the register has previously been 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
<a href="#">ID[63:32]</a>																															

### ID[63:32], bits [31:0]

PMCEID1[n] maps to event (n + 32). For a list of event numbers and descriptions, see 'Event numbers and mnemonics' in the ARM ARM, section D5.10.

For each bit:

ID[63:32]	Meaning
0	The common event is not implemented.
1	The common event is implemented.

Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture.

Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.

## Accessing the PMCEID1

PMCEID1 can be accessed through the external debug interface:

Component	Offset
PMU	0xE24

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# PMCEID2, Performance Monitors Common Event Identification register 2

The PMCEID2 characteristics are:

## Purpose

Defines which common architectural and common microarchitectural feature events in the range 0x4000 to 0x401F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

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:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<a href="#">ID[16415:16384]</a>																															

### ID[16415:16384], bits [31:0]

PMCEID2[31:0] maps to common events 0x4000 to 0x401F. For a list of event numbers and descriptions, see 'Event numbers and mnemonics' in the ARM ARM, section D5.10.

For each bit:

ID[16415:16384]	Meaning
0	The common event is not implemented.
1	The common event is implemented.

Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture.

Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.

## Accessing the PMCEID2

PMCEID2 can be accessed through the external debug interface:

Component	Offset
PMU	0xE28

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMCEID3, Performance Monitors Common Event Identification register 3

The PMCEID3 characteristics are:

## Purpose

Defines which common architectural and common microarchitectural feature events in the range 0x4020 to 0x403F are implemented. If a particular bit is set to 1, then the event for that bit is implemented.

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
<a href="#">ID[16447:16416]</a>																															

### ID[16447:16416], bits [31:0]

PMCEID3[31:0] maps to common events 0x4020 to 0x403F. For a list of event numbers and descriptions, see 'Event numbers and mnemonics' in the ARM ARM, section D5.10.

For each bit:

ID[16447:16416]	Meaning
0	The common event is not implemented.
1	The common event is implemented.

Bits that map to reserved event numbers are reserved to identify events that might be defined in future revisions to the architecture.

Events that do not require additional features in the PMU can be defined retrospectively, meaning that they can be implemented as part of a PMUv3 implementation.

## Accessing the PMCEID3

PMCEID3 can be accessed through the external debug interface:

Component	Offset
PMU	0xE2C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMCFGR, Performance Monitors Configuration Register

The PMCFGR characteristics are:

## Purpose

Contains PMU-specific configuration data.

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

PMCFGR is in the Core power domain.

## Attributes

PMCFGR is a 32-bit register.

## Field descriptions

The PMCFGR 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
NCG				0	0	0	0	0	0	0	0	UEN	WT	NA	EX	CCD	CC	SIZE					N								

### NCG, bits [31:28]

This feature is not supported, so this field is RAZ.

### Bits [27:20]

Reserved, RES0.

### UEN, bit [19]

User-mode Enable Register supported. [PMUSERENR\\_ELO](#) is not visible in the external debug interface, so this bit is RAZ.

### WT, bit [18]

This feature is not supported, so this bit is RAZ.

### NA, bit [17]

This feature is not supported, so this bit is RAZ.

**EX, bit [16]**

Export supported. Value is IMPLEMENTATION DEFINED.

EX	Meaning
0	<a href="#">PMCR_EL0.X</a> is RES0.
1	<a href="#">PMCR_EL0.X</a> is read/write.

**CCD, bit [15]**

Cycle counter has prescale. This is RES1 if AArch32 is supported at any EL, and RAZ otherwise.

CCD	Meaning
0	<a href="#">PMCR_EL0.D</a> is RES0.
1	<a href="#">PMCR_EL0.D</a> is read/write.

**CC, bit [14]**

Dedicated cycle counter (counter 31) supported. This bit is RAO.

**SIZE, bits [13:8]**

Size of counters. This field determines the spacing of counters in the memory-map.

In ARMv8 the counters are at doubleword-aligned addresses, and the largest counter is 64-bits, so this field is 0b111111.

**N, bits [7:0]**

Number of counters implemented in addition to the cycle counter, [PMCCNTR\\_EL0](#). The maximum number of event counters is 31.

N	Meaning
00000000	Only <a href="#">PMCCNTR_EL0</a> implemented.
00000001	<a href="#">PMCCNTR_EL0</a> plus one event counter implemented.

and so on up to 0b00011111, which indicates [PMCCNTR\\_EL0](#) and 31 event counters implemented.

**Accessing the PMCFGR**

PMCFGR can be accessed through the external debug interface:

Component	Offset
PMU	0xE00

# PMCID1SR, CONTEXTIDR\_EL1 Sample Register

The PMCID1SR characteristics are:

## Purpose

Contains the sampled value of [CONTEXTIDR\\_EL1](#), captured on reading [PMPCSR](#)[31:0].

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

## Configuration

PMCID1SR 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 if the OPTIONAL PC Sample-based Profiling Extension is implemented and ARMv8.2-PCSample is implemented. If the OPTIONAL PC Sample-based Profiling Extension is implemented and ARMv8.2-PCSample is not implemented, this register is not implemented and the architecture defines the functionality in [EDCIDS](#).

This register is introduced in ARMv8.2.

## Attributes

PMCID1SR is a 32-bit register.

## Field descriptions

The PMCID1SR 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
<a href="#">CONTEXTIDR_EL1</a>																															

### CONTEXTIDR\_EL1, bits [31:0]

Context ID. The value of CONTEXTIDR that is associated with the most recent [PMPCSR](#) sample.

- If EL1 is using AArch64, then the Context ID is held in [CONTEXTIDR\\_EL1](#).
- If EL1 is using AArch32, then the Context ID is held in [CONTEXTIDR](#).
- If EL3 is implemented and is using AArch32, then [CONTEXTIDR](#) is a banked register and PMCID1SR samples the current banked copy of [CONTEXTIDR](#) for the Security state that is associated with the most recent [PMPCSR](#) sample.

Because the value written to PMCID1SR is an indirect read of CONTEXTIDR, therefore it is CONSTRAINED UNPREDICTABLE whether PMCID1SR is set to the original or new value if a read of [PMPCSR](#) samples:

- An instruction that writes to [CONTEXTIDR\\_EL1](#).
- The next Context synchronization event.
- Any instruction executed between these two instructions.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

# Accessing the PMCID1SR

PMCID1SR can be accessed through the external debug interface:

Component	Offset
PMU	0x208
PMU	0x228

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# PMCID2SR, CONTEXTIDR\_EL2 Sample Register

The PMCID2SR characteristics are:

## Purpose

Contains the sampled value of [CONTEXTIDR\\_EL2](#), captured on reading [PMPCSR](#)[31:0].

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

## Configuration

PMCID2SR 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 if the OPTIONAL PC Sample-based Profiling Extension is implemented and ARMv8.2-PCSample is implemented. If the OPTIONAL PC Sample-based Profiling Extension is implemented and ARMv8.2-PCSample is not implemented, this register is not implemented and the architecture defines the functionality in [EDCIDS](#).

If EL2 is not implemented, this register is RES0.

This register is introduced in ARMv8.2.

## Attributes

PMCID2SR is a 32-bit register.

## Field descriptions

The PMCID2SR 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
<a href="#">CONTEXTIDR_EL2</a>																															

### CONTEXTIDR\_EL2, bits [31:0]

Context ID. The value of CONTEXTIDR that is associated with the most recent [PMPCSR](#) sample.

- If EL2 is using AArch64, then the Context ID is in [CONTEXTIDR\\_EL2](#).
- If EL2 is using AArch32, then this field is set to an UNKNOWN value.
- If [PMPCSR](#).NS==0, this field is set to an UNKNOWN value.

Because the value written to PMCID2SR is an indirect read of CONTEXTIDR, therefore it is CONSTRAINED UNPREDICTABLE whether PMCID2SR is set to the original or new value if a read of [PMPCSR](#) samples:

- An instruction that writes to [CONTEXTIDR\\_EL2](#).
- The next Context synchronization event.
- Any instruction executed between these two instructions.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the PMCID2SR

PMCID2SR can be accessed through the external debug interface:

Component	Offset
PMU	0x22C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMCIDR0, Performance Monitors Component Identification Register 0

The PMCIDR0 characteristics are:

## Purpose

Provides information to identify a Performance Monitor component.

For more information see 'About the Component identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMCIDR0 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

PMCIDR0 is a 32-bit register.

## Field descriptions

The PMCIDR0 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								
																								PRMBL_0							

### Bits [31:8]

Reserved, RES0.

### PRMBL\_0, bits [7:0]

Preamble. Must read as 0x0D.

## Accessing the PMCIDR0

PMCIDR0 can be accessed through the external debug interface:

Component	Offset
PMU	0xFF0

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMCIDR1, Performance Monitors Component Identification Register 1

The PMCIDR1 characteristics are:

## Purpose

Provides information to identify a Performance Monitor component.

For more information see 'About the Component identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMCIDR1 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

PMCIDR1 is a 32-bit register.

## Field descriptions

The PMCIDR1 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	CLASS				PRMBL_1			

### Bits [31:8]

Reserved, RES0.

### CLASS, bits [7:4]

Component class. Reads as 0x9, debug component.

### PRMBL\_1, bits [3:0]

Preamble. RAZ.

## Accessing the PMCIDR1

PMCIDR1 can be accessed through the external debug interface:

Component	Offset
PMU	0xFF4

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMCIDR2, Performance Monitors Component Identification Register 2

The PMCIDR2 characteristics are:

## Purpose

Provides information to identify a Performance Monitor component.

For more information see 'About the Component identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMCIDR2 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

PMCIDR2 is a 32-bit register.

## Field descriptions

The PMCIDR2 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	PRMBL_2										

### Bits [31:8]

Reserved, RES0.

### PRMBL\_2, bits [7:0]

Preamble. Must read as 0x05.

## Accessing the PMCIDR2

PMCIDR2 can be accessed through the external debug interface:

Component	Offset
PMU	0xFF8

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.



# PMCIDR3, Performance Monitors Component Identification Register 3

The PMCIDR3 characteristics are:

## Purpose

Provides information to identify a Performance Monitor component.

For more information see 'About the Component identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMCIDR3 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

PMCIDR3 is a 32-bit register.

## Field descriptions

The PMCIDR3 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	PRMBL 3									

### Bits [31:8]

Reserved, RES0.

### PRMBL\_3, bits [7:0]

Preamble. Must read as 0xB1.

## Accessing the PMCIDR3

PMCIDR3 can be accessed through the external debug interface:

Component	Offset
PMU	0xFFC

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMCNTENCLR\_EL0, Performance Monitors Count Enable Clear register

The PMCNTENCLR\_EL0 characteristics are:

## Purpose

Disables the Cycle Count Register, [PMCCNTR\\_EL0](#), and any implemented event counters [PMEVCNTR<n>](#). Reading this register shows which counters are enabled.

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	RW

## Configuration

External register PMCNTENCLR\_EL0 is architecturally mapped to AArch64 System register [PMCNTENCLR\\_EL0](#).

External register PMCNTENCLR\_EL0 is architecturally mapped to AArch32 System register [PMCNTENCLR](#).

PMCNTENCLR\_EL0 is in the Core power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply on a Warm or Cold reset. The register is not affected by an External debug reset.

## Attributes

PMCNTENCLR\_EL0 is a 32-bit register.

## Field descriptions

The PMCNTENCLR\_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
C	P<n>, bit [n]																														

### C, bit [31]

[PMCCNTR\\_EL0](#) 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.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### P<n>, bit [n], for n = 0 to 30

Event counter disable bit for [PMEVCNTR<n>\\_EL0](#).

Bits [30:N] are RAZ/WI. N is the value in [PMCFGR.N](#).

Possible values of each bit are:

P<n>	Meaning
0	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> is disabled. When written, has no effect.
1	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> is enabled. When written, disables <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the PMCNTENCLR\_EL0

PMCNTENCLR\_EL0 can be accessed through the external debug interface:

Component	Offset
PMU	0xC20

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMCNTENSET\_EL0, Performance Monitors Count Enable Set register

The PMCNTENSET\_EL0 characteristics are:

## Purpose

Enables the Cycle Count Register, [PMCCNTR\\_EL0](#), and any implemented event counters [PMEVCNTR<n>](#). Reading this register shows which counters are enabled.

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	RW

## Configuration

External register PMCNTENSET\_EL0 is architecturally mapped to AArch64 System register [PMCNTENSET\\_EL0](#).

External register PMCNTENSET\_EL0 is architecturally mapped to AArch32 System register [PMCNTENSET](#).

PMCNTENSET\_EL0 is in the Core power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply on a Warm or Cold reset. The register is not affected by an External debug reset.

## Attributes

PMCNTENSET\_EL0 is a 32-bit register.

## Field descriptions

The PMCNTENSET\_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
C	P<n>, bit [n]																														

### C, bit [31]

[PMCCNTR\\_EL0](#) 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.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### P<n>, bit [n], for n = 0 to 30

Event counter enable bit for [PMEVCNTR<n>\\_EL0](#).

Bits [30:N] are RAZ/WI. N is the value in [PMCFGR.N](#).

Possible values of each bit are:

P<n>	Meaning
0	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> is disabled. When written, has no effect.
1	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> event counter is enabled. When written, enables <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> .

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the PMCNTENSET\_EL0

PMCNTENSET\_EL0 can be accessed through the external debug interface:

Component	Offset
PMU	0xC00

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMCR\_EL0, Performance Monitors Control Register

The PMCR\_EL0 characteristics are:

## Purpose

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

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	RW

## Configuration

External register PMCR\_EL0 bits [6:0] are architecturally mapped to AArch32 System register [PMCR\[6:0\]](#).

External register PMCR\_EL0 bits [6:0] are architecturally mapped to AArch64 System register [PMCR\\_EL0\[6:0\]](#).

PMCR\_EL0 is in the Core power domain. Some or all RW fields of this register have defined reset values. These apply on a Warm or Cold reset. The register is not affected by an External debug reset.

This register is only partially mapped to the internal [PMCR](#) System register. An external agent must use other means to discover the information held in [PMCR\[31:11\]](#), such as accessing [PMCFGR](#) and the ID registers.

## Attributes

PMCR\_EL0 is a 32-bit register.

## Field descriptions

The PMCR\_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	LC	DP	X	D	C	P	E

### Bits [31:11]

RAZ/WI. Hardware must implement this field as RAZ/WI. Software must not rely on the register reading as zero, and must use a read-modify-write sequence to write to the register.

### Bits [10:7]

Reserved, RES0.

### LC, bit [6]

Long cycle counter enable. Determines which [PMCCNTR\\_EL0](#) bit generates an overflow recorded by [PMOVSr\[31\]](#).

LC	Meaning
0	Cycle counter overflow on increment that changes <a href="#">PMCCNTR_EL0</a> [31] from 1 to 0.
1	Cycle counter overflow on increment that changes <a href="#">PMCCNTR_EL0</a> [63] from 1 to 0.

ARM deprecates use of PMCR\_EL0.LC = 0.

In an AArch64-only implementation, this field 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.

#### DP, bit [5]

Disable cycle counter when event counting is prohibited. The possible values of this bit are:

DP	Meaning
0	<a href="#">PMCCNTR_EL0</a> , if enabled, counts when event counting is prohibited.
1	<a href="#">PMCCNTR_EL0</a> 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:

- A value that is architecturally UNKNOWN if the reset is into an Exception level that is using AArch64.
- 0 if the reset is into an Exception level that is using AArch32.

#### 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:

- A value that is architecturally UNKNOWN if the reset is into an Exception level that is using AArch64.
- 0 if the reset is into an Exception level that is using AArch32.

#### D, bit [3]

Clock divider. The possible values of this bit are:

D	Meaning
0	When enabled, <a href="#">PMCCNTR_EL0</a> counts every clock cycle.
1	When enabled, <a href="#">PMCCNTR_EL0</a> counts once every 64 clock cycles.

In an AArch64-only implementation this field is RES0, otherwise it is an RW field. If PMCR\_EL0.LC == 1, this bit is ignored and the cycle counter counts every clock cycle.

ARM deprecates use of PMCR\_EL0.D = 1.



When this register has an architecturally-defined reset value, if this field is implemented as an RW field it resets to:

- A value that is architecturally UNKNOWN if the reset is into an Exception level that is using AArch64.
- 0 if the reset is into an Exception level that is using AArch32.

### 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 <a href="#">PMCCNTR_EL0</a> to zero.

This bit is always RAZ.

Resetting [PMCCNTR\\_EL0](#) does not clear the [PMCCNTR\\_EL0](#) 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, not including <a href="#">PMCCNTR_EL0</a> , to zero.

This bit is always RAZ.

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, including <a href="#">PMCCNTR_EL0</a> , are disabled.
1	All counters are enabled by <a href="#">PMCNTENSET_EL0</a> .

This bit is RW.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the PMCR\_EL0

PMCR\_EL0 can be accessed through the external debug interface:

Component	Offset
PMU	0xE04

# PMDEVAFF0, Performance Monitors Device Affinity register 0

The PMDEVAFF0 characteristics are:

## Purpose

Copy of the low half of the PE [MPIDR\\_EL1](#) register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMDEVAFF0 is in the Debug power domain.

Implementation of this register is OPTIONAL.

## Attributes

PMDEVAFF0 is a 32-bit register.

## Field descriptions

The PMDEVAFF0 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
																<a href="#">MPIDR_EL1 low half</a>															

### Bits [31:0]

[MPIDR\\_EL1](#) low half. Read-only copy of the low half of [MPIDR\\_EL1](#), as seen from the highest implemented Exception level.

## Accessing the PMDEVAFF0

PMDEVAFF0 can be accessed through the external debug interface:

Component	Offset
PMU	0xFA8

# PMDEVAFF1, Performance Monitors Device Affinity register 1

The PMDEVAFF1 characteristics are:

## Purpose

Copy of the high half of the PE [MPIDR\\_EL1](#) register that allows a debugger to determine which PE in a multiprocessor system the Performance Monitor component relates to.

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMDEVAFF1 is in the Debug power domain.

Implementation of this register is OPTIONAL.

## Attributes

PMDEVAFF1 is a 32-bit register.

## Field descriptions

The PMDEVAFF1 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
																		<a href="#">MPIDR_EL1 high half</a>													

### Bits [31:0]

[MPIDR\\_EL1](#) high half. Read-only copy of the high half of [MPIDR\\_EL1](#), as seen from the highest implemented Exception level.

## Accessing the PMDEVAFF1

PMDEVAFF1 can be accessed through the external debug interface:

Component	Offset
PMU	0xFAC

# PMDEVARCH, Performance Monitors Device Architecture register

The PMDEVARCH characteristics are:

## Purpose

Identifies the programmers' model architecture of the Performance Monitor component.

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMDEVARCH is in the Debug power domain.

## Attributes

PMDEVARCH is a 32-bit register.

## Field descriptions

The PMDEVARCH 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
ARCHITECT											PRESENT	REVISION				ARCHID															

### ARCHITECT, bits [31:21]

Defines the architecture of the component. For Performance Monitors, this is ARM Limited.

Bits [31:28] are the JEP106 continuation code,  $0 \times 4$ .

Bits [27:21] are the JEP106 ID code,  $0 \times 3B$ .

### PRESENT, bit [20]

When set to 1, indicates that the DEVARCH is present.

This field is 1 in ARMv8.

### REVISION, bits [19:16]

Defines the architecture revision. For architectures defined by ARM this is the minor revision.

For Performance Monitors, the revision defined by ARMv8 is  $0 \times 0$ .

All other values are reserved.

**ARCHID, bits [15:0]**

Defines this part to be an ARMv8 debug component. For architectures defined by ARM this is further subdivided.

For Performance Monitors:

- Bits [15:12] are the architecture version, 0x2.
- Bits [11:0] are the architecture part number, 0xA16.

This corresponds to Performance Monitors architecture version PMUv3.

---

**Note**

The PMUv3 memory-mapped programmers' model can be used by devices other than ARMv8 processors. Software must determine whether the PMU is attached to an ARMv8 processor by using the [PMDEVAFF0](#) and [PMDEVAFF1](#) registers to discover the affinity of the PMU to any ARMv8 processors.

---

## Accessing the PMDEVARCH

PMDEVARCH can be accessed through the external debug interface:

Component	Offset
PMU	0xFBC

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMDEVID, Performance Monitors Device ID register

The PMDEVID characteristics are:

## Purpose

Provides information about features of the debug implementation.

This register is part of the Performance Monitors registers functional group.

## Usage constraints

Accessing this register depends on which field is being accessed; see the register field descriptions for the states that they are accessible in.

## Configuration

PMDEVID is in the Debug power domain.

This register is required in all implementations.

Up until and including ARMv8.1, the architecture defines the functionality in a different set of registers, see [DBGDEVID](#).

## Attributes

PMDEVID is a 32-bit register.

## Field descriptions

The PMDEVID 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	PCSample

### Bits [31:4]

Reserved, RES0.

### PCSample, bits [3:0]

Indicates the level of PC Sample-based Profiling support using performance monitor registers. Permitted values of this field are:

PCSample	Meaning
0000	Architecture-defined form of PC Sample-based Profiling not implemented with performance monitor registers.
0001	Architecture-defined form of PC Sample-based Profiling is implemented with performance monitor registers.

All other values are reserved.

## Accessing the PMDEVID

PMDEVID can be accessed through the external debug interface:

Component	Offset
PMU	0xFC8



# PMDEVTYPE, Performance Monitors Device Type register

The PMDEVTYPE characteristics are:

## Purpose

Indicates to a debugger that this component is part of a PE's performance monitor interface.

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMDEVTYPE is in the Debug power domain.

Implementation of this register is OPTIONAL.

## Attributes

PMDEVTYPE is a 32-bit register.

## Field descriptions

The PMDEVTYPE 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	SUB				MAJOR			

### Bits [31:8]

Reserved, RES0.

### SUB, bits [7:4]

Subtype. Must read as 0x1 to indicate this is a component within a PE.

### MAJOR, bits [3:0]

Major type. Must read as 0x6 to indicate this is a performance monitor component.

## Accessing the PMDEVTYPE

PMDEVTYPE can be accessed through the external debug interface:

Component	Offset
PMU	0xFCC





# PMEVCNTR<n>\_EL0, Performance Monitors Event Count Registers, n = 0 - 30

The PMEVCNTR<n>\_EL0 characteristics are:

## Purpose

Holds event counter n, which counts events, where n is 0 to 30.

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	RW

External accesses to the performance monitors ignore [PMUSERENR\\_EL0](#) and, if implemented, [MDCR\\_EL2](#). {TPM, TPMCR, HPMN} and [MDCR\\_EL3](#). TPM. This means that all counters are accessible regardless of the current EL or privilege of the access.

## Configuration

External register PMEVCNTR<n>\_EL0 is architecturally mapped to AArch64 System register [PMEVCNTR<n>\\_EL0](#).

External register PMEVCNTR<n>\_EL0 is architecturally mapped to AArch32 System register [PMEVCNTR<n>](#).

PMEVCNTR<n>\_EL0 is in the Core power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply on a Warm or Cold reset. The register is not affected by an External debug reset.

## Attributes

PMEVCNTR<n>\_EL0 is a 32-bit register.

## Field descriptions

The PMEVCNTR<n>\_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
Event counter n																															

### Bits [31:0]

Event counter n. Value of event counter n, where n is the number of this register and is a number from 0 to 30.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the PMEVCNTR<n>\_EL0

PMEVCNTR<n>\_EL0 can be accessed through the external debug interface:

Component	Offset
PMU	0x000 + 8n



# PMEVTYPER<n>\_EL0, Performance Monitors Event Type Registers, n = 0 - 30

The PMEVTYPER<n>\_EL0 characteristics are:

## Purpose

Configures event counter n, where n is 0 to 30.

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	RW

## Configuration

External register PMEVTYPER<n>\_EL0 is architecturally mapped to AArch64 System register [PMEVTYPER<n>\\_EL0](#).

External register PMEVTYPER<n>\_EL0 is architecturally mapped to AArch32 System register [PMEVTYPER<n>](#).

PMEVTYPER<n>\_EL0 is in the Core power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply on a Warm or Cold reset. The register is not affected by an External debug reset.

## Attributes

PMEVTYPER<n>\_EL0 is a 32-bit register.

## Field descriptions

The PMEVTYPER<n>\_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
P	U	NSK	NSU	NSH	M	MT	0	0	0	0	0	0	0	0	0	evtCount[15:10]						evtCount[9:0]									

### P, bit [31]

Privileged filtering bit. Controls counting in EL1. If EL3 is implemented, then counting in Non-secure EL1 is further controlled by the NSK bit. The possible values of this bit are:

P	Meaning
0	Count events in EL1.
1	Do not count events in EL1.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### U, bit [30]

User filtering bit. Controls counting in EL0. If EL3 is implemented, then counting in Non-secure EL0 is further controlled by the NSU bit. The possible values of this bit are:

U	Meaning
0	Count events in EL0.
1	Do not count events in EL0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### NSK, bit [29]

Non-secure EL1 (kernel) modes filtering bit. Controls counting in Non-secure EL1. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of P, events in Non-secure EL1 are counted.

Otherwise, events in Non-secure EL1 are not counted.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### NSU, bit [28]

Non-secure EL0 (Unprivileged) filtering. Controls counting in Non-secure EL0. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of U, events in Non-secure EL0 are counted.

Otherwise, events in Non-secure EL0 are not counted.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### NSH, bit [27]

Non-secure EL2 (Hypervisor) filtering. Controls counting in Non-secure EL2. If EL2 is not implemented, this bit is RES0.

NSH	Meaning
0	Do not count events in EL2.
1	Count events in EL2.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### M, bit [26]

Secure EL3 filtering bit. If EL3 is not implemented, this bit is RES0.

If the value of this bit is equal to the value of P, cycles in Secure EL3 are counted.

Otherwise, cycles in Secure EL3 are not counted.

Most applications can ignore this field and set its value to 0.

#### Note

This field is not visible in the AArch32 PMEVTYPER System register.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

#### MT, bit [25]

Multithreading. When the implementation is multi-threaded, the valid values for this bit are:

MT	Meaning
0	Count events only on controlling PE.
1	Count events from any PE with the same affinity at level 1 and above as this PE.

When the implementation is not multi-threaded, this bit is RES0.

#### Note

- An implementation is described as multi-threaded when the lowest level of affinity consists of logical PEs that are implemented using a multi-threading type approach. That is, the performance of PEs at the lowest affinity level is highly interdependent. On such an implementation, the value of MPIDR\_EL1.MT, when read at the highest implemented Exception level, is 1.
- Events from a different thread of a multithreaded implementation are not Attributable to the thread counting the event.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Bits [24:16]**

Reserved, RES0.

**evtCount[15:10], bits [15:10]**  
**In ARMv8.2 and ARMv8.1:**

Extension to evtCount[9:0]. See evtCount[9:0] 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.

**evtCount[9:0], bits [9:0]**

Event to count. The event number of the event that is counted by event counter [PMEVCNTR<n>\\_EL0](#).

Software must program this field with an event that is supported by the PE being programmed.

There are three ranges of event numbers:

- Event numbers in the range 0x000 to 0x03F are common architectural and microarchitectural events.
- Event numbers in the range 0x040 to 0x0BF are ARM recommended common architectural and microarchitectural events.
- Event numbers in the range 0x0C0 to 0x3FF are IMPLEMENTATION DEFINED events.

If evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the event type:

- For the range 0x000 to 0x03F, no events are counted, and the value returned by a direct or external read of the evtCount field is the value written to the field.
- For IMPLEMENTATION DEFINED events, it is UNPREDICTABLE what event, if any, is counted, and the value returned by a direct or external read of the evtCount field is UNKNOWN.

**Note**

UNPREDICTABLE means the event must not expose privileged information.

ARM recommends that the behavior across a family of implementations is defined such that if a given implementation does not include an event from a set of common IMPLEMENTATION DEFINED events, then no event is counted and the value read back on evtCount is the value written.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

**Accessing the PMEVTYPER<n>\_EL0**

PMEVTYPER<n>\_EL0 can be accessed through the external debug interface:

Component	Offset
PMU	0x400 + 4n



# PMINTENCLR\_EL1, Performance Monitors Interrupt Enable Clear register

The PMINTENCLR\_EL1 characteristics are:

## Purpose

Disables the generation of interrupt requests on overflows from the Cycle Count Register, [PMCCNTR\\_EL0](#), and the event counters [PMEVCNTR<n>\\_EL0](#). Reading the register shows which overflow interrupt requests are enabled.

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EPMA	SLK	Default
Error	Error	Error	Error	RO	RW

## Configuration

External register PMINTENCLR\_EL1 is architecturally mapped to AArch64 System register [PMINTENCLR\\_EL1](#).

External register PMINTENCLR\_EL1 is architecturally mapped to AArch32 System register [PMINTENCLR](#).

PMINTENCLR\_EL1 is in the Core power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply on a Warm or Cold reset. The register is not affected by an External debug reset.

## Attributes

PMINTENCLR\_EL1 is a 32-bit register.

## Field descriptions

The PMINTENCLR\_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
C	P<n>, bit [n]																														

### C, bit [31]

[PMCCNTR\\_EL0](#) 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.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### P<n>, bit [n], for n = 0 to 30

Event counter overflow interrupt request disable bit for [PMEVCNTR<n>\\_EL0](#).

Bits [30:N] are RAZ/WI. N is the value in [PMCFGR.N](#).



Possible values are:

P<n>	Meaning
0	When read, means that the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> event counter interrupt request is disabled. When written, has no effect.
1	When read, means that the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> event counter interrupt request is enabled. When written, disables the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> interrupt request.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the PMINTENCLR\_EL1

PMINTENCLR\_EL1 can be accessed through the external debug interface:

Component	Offset
PMU	0xC60

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMINTENSET\_EL1, Performance Monitors Interrupt Enable Set register

The PMINTENSET\_EL1 characteristics are:

## Purpose

Enables the generation of interrupt requests on overflows from the Cycle Count Register, [PMCCNTR\\_EL0](#), and the event counters [PMEVCNTR<n>\\_EL0](#). Reading the register shows which overflow interrupt requests are enabled.

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	RW

## Configuration

External register PMINTENSET\_EL1 is architecturally mapped to AArch64 System register [PMINTENSET\\_EL1](#).

External register PMINTENSET\_EL1 is architecturally mapped to AArch32 System register [PMINTENSET](#).

PMINTENSET\_EL1 is in the Core power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply on a Warm or Cold reset. The register is not affected by an External debug reset.

## Attributes

PMINTENSET\_EL1 is a 32-bit register.

## Field descriptions

The PMINTENSET\_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
C	P<n>, bit [n]																														

### C, bit [31]

[PMCCNTR\\_EL0](#) 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.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### P<n>, bit [n], for n = 0 to 30

Event counter overflow interrupt request enable bit for [PMEVCNTR<n>\\_EL0](#).

Bits [30:N] are RAZ/WI. N is the value in [PMCFGR.N](#).

Possible values are:

P<n>	Meaning
0	When read, means that the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> event counter interrupt request is disabled. When written, has no effect.
1	When read, means that the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> event counter interrupt request is enabled. When written, enables the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> interrupt request.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the PMINTENSET\_EL1

PMINTENSET\_EL1 can be accessed through the external debug interface:

Component	Offset
PMU	0xC40

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMITCTRL, Performance Monitors Integration mode Control register

The PMITCTRL characteristics are:

## Purpose

Enables the Performance Monitors to switch from default mode into integration mode, where test software can control directly the inputs and outputs of the PE, for integration testing or topology detection.

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	Default
IMP DEF	IMP DEF	IMP DEF	RW

## Configuration

It is IMPLEMENTATION DEFINED whether PMITCTRL is implemented in the Core power domain or in the Debug power domain. Some or all RW fields of this register have defined reset values, and:

- The register is not affected by a Warm reset.
- If the register is implemented in the Core power domain the reset values apply on a Cold reset, and the register is not affected by an External debug reset.
- If the register is implemented in the Debug power domain the reset values apply on an External debug reset, and the register is not affected by a Cold reset.

Implementation of this register is OPTIONAL.

## Attributes

PMITCTRL is a 32-bit register.

## Field descriptions

The PMITCTRL 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	IME

### Bits [31:1]

Reserved, RES0.

### IME, bit [0]

Integration mode enable. When IME == 1, the device reverts to an integration mode to enable integration testing or topology detection. The integration mode behavior is IMPLEMENTATION DEFINED.

IME	Meaning
0	Normal operation.
1	Integration mode enabled.

When this register has an architecturally-defined reset value, this field resets to 0.

## Accessing the PMITCTRL

PMITCTRL can be accessed through the external debug interface:

Component	Offset
PMU	0xF00

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMLAR, Performance Monitors Lock Access Register

The PMLAR characteristics are:

## Purpose

Allows or disallows access to the Performance Monitors registers through a memory-mapped interface.

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

Default
WO

## Configuration

PMLAR is in the Debug power domain.

If OPTIONAL memory-mapped access to the external debug interface is supported then an OPTIONAL Software Lock can be implemented as part of CoreSight compliance.

PMLAR ignores writes if the Software Lock is not implemented and ignores writes for other accesses to the external debug interface.

The Software Lock provides a lock to prevent memory-mapped writes to the Performance Monitors registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Performance Monitors registers. It does not, and cannot, prevent all accidental or malicious damage.

Software uses PMLAR to set or clear the lock, and [PMLSR](#) to check the current status of the lock.

## Attributes

PMLAR is a 32-bit register.

## Field descriptions

The PMLAR 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
																KEY															

### KEY, bits [31:0]

Lock Access control. Writing the key value 0xC5ACCE55 to this field unlocks the lock, enabling write accesses to this component's registers through a memory-mapped interface.

Writing any other value to this register locks the lock, disabling write accesses to this component's registers through a memory mapped interface.

## Accessing the PMLAR

PMLAR can be accessed through a memory-mapped access to the external debug interface:

Component	Offset
PMU	0xFB0



# PMLSR, Performance Monitors Lock Status Register

The PMLSR characteristics are:

## Purpose

Indicates the current status of the software lock for Performance Monitors registers.

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

Default
RO

## Configuration

PMLSR is in the Debug power domain. Some or all RW fields of this register have defined reset values. These apply only on an External debug reset. The register is not affected by a Warm reset and is not affected by a Cold reset.

If OPTIONAL memory-mapped access to the external debug interface is supported then an OPTIONAL Software Lock can be implemented as part of CoreSight compliance.

PMLSR is RAZ if the Software Lock is not implemented and is RAZ for other accesses to the external debug interface.

The Software Lock provides a lock to prevent memory-mapped writes to the Performance Monitors registers. Use of this lock mechanism reduces the risk of accidental damage to the contents of the Performance Monitors registers. It does not, and cannot, prevent all accidental or malicious damage.

Software uses [PMLAR](#) to set or clear the lock, and PMLSR to check the current status of the lock.

## Attributes

PMLSR is a 32-bit register.

## Field descriptions

The PMLSR 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	nTT	SLK	SLI

### Bits [31:3]

Reserved, RES0.

### nTT, bit [2]

Not thirty-two bit access required. RAZ.

### SLK, bit [1]

Software Lock status for this component. For an access to LSR that is not a memory-mapped access, or when the Software Lock is not implemented, this field is RES0.



For memory-mapped accesses when the software lock is implemented, possible values of this field are:

SLK	Meaning
0	Lock clear. Writes are permitted to this component's registers.
1	Lock set. Writes to this component's registers are ignored, and reads have no side effects.

When this register has an architecturally-defined reset value, this field resets to 1.

### SLI, bit [0]

Software Lock implemented. For an access to LSR that is not a memory-mapped access, this field is RAZ. For memory-mapped accesses, the value of this field is IMPLEMENTATION DEFINED. Permitted values are:

SLI	Meaning
0	Software Lock not implemented or not memory-mapped access.
1	Software Lock implemented and memory-mapped access.

## Accessing the PMLSR

PMLSR can be accessed through a memory-mapped access to the external debug interface:

Component	Offset
PMU	0xFB4

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMOVSLR\_EL0, Performance Monitors Overflow Flag Status Clear register

The PMOVSLR\_EL0 characteristics are:

## Purpose

Contains the state of the overflow bit for the Cycle Count Register, [PMCCNTR\\_EL0](#), and each of the implemented event counters [PMEVCNTR<n>](#). Writing to this register clears these bits.

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

Off	DLK	OSLK	EPMA	SLK	Default
Error	Error	Error	Error	RO	RW

## Configuration

External register PMOVSLR\_EL0 is architecturally mapped to AArch64 System register [PMOVSLR\\_EL0](#).

External register PMOVSLR\_EL0 is architecturally mapped to AArch32 System register [PMOVSRR](#).

PMOVSLR\_EL0 is in the Core power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply on a Warm or Cold reset. The register is not affected by an External debug reset.

## Attributes

PMOVSLR\_EL0 is a 32-bit register.

## Field descriptions

The PMOVSLR\_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
C	P<n>, bit [n]																														

### C, bit [31]

[PMCCNTR\\_EL0](#) 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\\_EL0](#).LC controls whether an overflow is detected from [PMCCNTR\\_EL0](#)[31] or from [PMCCNTR\\_EL0](#)[63].

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### P<n>, bit [n], for n = 0 to 30

Event counter overflow clear bit for [PMEVCNTR<n>\\_EL0](#).

Bits [30:N] are RAZ/WI. N is the value in [PMCFGR.N](#).

Possible values of each bit are:

P<n>	Meaning
0	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> has not overflowed. When written, has no effect.
1	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> has overflowed. When written, clears the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> overflow bit to 0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the PMOVSLR\_EL0

PMOVSLR\_EL0 can be accessed through the external debug interface:

Component	Offset
PMU	0xC80

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMOVSSET\_EL0, Performance Monitors Overflow Flag Status Set register

The PMOVSSET\_EL0 characteristics are:

## Purpose

Sets the state of the overflow bit for the Cycle Count Register, [PMCCNTR\\_EL0](#), and each of the implemented event counters [PMEVCNTR<n>](#).

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	RW

## Configuration

External register PMOVSSET\_EL0 is architecturally mapped to AArch64 System register [PMOVSSET\\_EL0](#).

External register PMOVSSET\_EL0 is architecturally mapped to AArch32 System register [PMOVSSET](#).

PMOVSSET\_EL0 is in the Core power domain. RW fields in this register reset to architecturally UNKNOWN values. These apply on a Warm or Cold reset. The register is not affected by an External debug reset.

## Attributes

PMOVSSET\_EL0 is a 32-bit register.

## Field descriptions

The PMOVSSET\_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
C	P<n>, bit [n]																														

### C, bit [31]

[PMCCNTR\\_EL0](#) 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.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

### P<n>, bit [n], for n = 0 to 30

Event counter overflow set bit for [PMEVCNTR<n>\\_EL0](#).

Bits [30:N] are RAZ/WI. N is the value in [PMCFGR.N](#).

Possible values are:

P<n>	Meaning
0	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> has not overflowed. When written, has no effect.
1	When read, means that <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> has overflowed. When written, sets the <a href="#">PMEVCNTR&lt;n&gt;_EL0</a> overflow bit to 1.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the PMOVSSET\_EL0

PMOVSSET\_EL0 can be accessed through the external debug interface:

Component	Offset
PMU	0xCC0

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# 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

## 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 if the OPTIONAL PC Sample-based Profiling Extension is implemented and ARMv8.2-PCSample is implemented.

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.

If the OPTIONAL PC Sample-based Profiling Extension is implemented and ARMv8.2-PCSample is not implemented, this register is not implemented and the architecture defines the functionality in [EDPCSR](#).

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	PC Sample[55:32]																								
PC Sample[31:0]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

### NS, bit [63]

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.

**PC Sample[31:0], bits [31:0]**

Bits[31:0] of the sampled instruction address value. The translation regime that PMPCSR samples can be determined from PMPCSR.{NS,EL}.

- For a read of PMPCSR[31:0] from the memory-mapped interface, if PMLSR.SLK == 1, meaning the OPTIONAL Software Lock is locked, then the 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.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally 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

# PMPIDR0, Performance Monitors Peripheral Identification Register 0

The PMPIDR0 characteristics are:

## Purpose

Provides information to identify a Performance Monitor component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMPIDR0 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

PMPIDR0 is a 32-bit register.

## Field descriptions

The PMPIDR0 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								

[PART\\_0](#)

### Bits [31:8]

Reserved, RES0.

### PART\_0, bits [7:0]

Part number, least significant byte.

## Accessing the PMPIDR0

PMPIDR0 can be accessed through the external debug interface:

Component	Offset
PMU	0xFE0





# PMPIDR1, Performance Monitors Peripheral Identification Register 1

The PMPIDR1 characteristics are:

## Purpose

Provides information to identify a Performance Monitor component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMPIDR1 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

PMPIDR1 is a 32-bit register.

## Field descriptions

The PMPIDR1 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	DES_0				PART_1			

### Bits [31:8]

Reserved, RES0.

### DES\_0, bits [7:4]

Designer, least significant nibble of JEP106 ID code. For ARM Limited, this field is 0b1011.

### PART\_1, bits [3:0]

Part number, most significant nibble.

## Accessing the PMPIDR1

PMPIDR1 can be accessed through the external debug interface:

Component	Offset
PMU	0xFE4

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMPIDR2, Performance Monitors Peripheral Identification Register 2

The PMPIDR2 characteristics are:

## Purpose

Provides information to identify a Performance Monitor component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMPIDR2 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

PMPIDR2 is a 32-bit register.

## Field descriptions

The PMPIDR2 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	REVISION	JEDEC	DES_1					

### Bits [31:8]

Reserved, RES0.

### REVISION, bits [7:4]

Part major revision. Parts can also use this field to extend Part number to 16-bits.

### JEDEC, bit [3]

RAO. Indicates a JEP106 identity code is used.

### DES\_1, bits [2:0]

Designer, most significant bits of JEP106 ID code. For ARM Limited, this field is 0b011.

## Accessing the PMPIDR2

PMPIDR2 can be accessed through the external debug interface:

Component	Offset
PMU	0xFE8

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMPIDR3, Performance Monitors Peripheral Identification Register 3

The PMPIDR3 characteristics are:

## Purpose

Provides information to identify a Performance Monitor component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMPIDR3 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

PMPIDR3 is a 32-bit register.

## Field descriptions

The PMPIDR3 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	0

### Bits [31:8]

Reserved, RES0.

### REVAND, bits [7:4]

Part minor revision. Parts using [PMPIDR2](#).REVISION as an extension to the Part number must use this field as a major revision number.

### CMOD, bits [3:0]

Customer modified. Indicates someone other than the Designer has modified the component.

## Accessing the PMPIDR3

PMPIDR3 can be accessed through the external debug interface:

Component	Offset
PMU	0xFEC

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMPIDR4, Performance Monitors Peripheral Identification Register 4

The PMPIDR4 characteristics are:

## Purpose

Provides information to identify a Performance Monitor component.

For more information see 'About the Peripheral identification scheme' in the ARMv8 ARM, section H8 (About the External Debug Registers).

This register is part of the Performance Monitors registers functional group.

## Usage constraints

This register is accessible as follows:

SLK	Default
RO	RO

## Configuration

PMPIDR4 is in the Debug power domain.

Implementation of this register is OPTIONAL.

This register is required for CoreSight compliance.

## Attributes

PMPIDR4 is a 32-bit register.

## Field descriptions

The PMPIDR4 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		SIZE				DES_2		

### Bits [31:8]

Reserved, RES0.

### SIZE, bits [7:4]

Size of the component. RAZ. Log2 of the number of 4KB pages from the start of the component to the end of the component ID registers.

### DES\_2, bits [3:0]

Designer, JEP106 continuation code, least significant nibble. For ARM Limited, this field is 0b0100.

## Accessing the PMPIDR4

PMPIDR4 can be accessed through the external debug interface:



Component	Offset
PMU	0xFD0

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMSWINC\_EL0, Performance Monitors Software Increment register

The PMSWINC\_EL0 characteristics are:

## Purpose

Increments a counter that is configured to count the Software increment event, event 0x00. For more information, see 'SW\_INCR' in the ARMv8 ARM, section D5.

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	WI	WO

## Configuration

External register PMSWINC\_EL0 is architecturally mapped to AArch64 System register [PMSWINC\\_EL0](#).

External register PMSWINC\_EL0 is architecturally mapped to AArch32 System register [PMSWINC](#).

PMSWINC\_EL0 is in the Core power domain.

Implementation of this register is OPTIONAL.

If this register is implemented, use of it is deprecated.

If 1 is written to bit [n] from the external debug interface, it is CONSTRAINED UNPREDICTABLE whether or not a SW\_INCR event is created for counter n. This is consistent with not implementing the register in the external debug interface.

## Attributes

PMSWINC\_EL0 is a 32-bit register.

## Field descriptions

The PMSWINC\_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	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>\\_EL0](#).

P<n> is WI if n >= [PMCR\\_EL0](#).N, the number of implemented counters.

Otherwise, the effects of writing to this bit are:

P<n>	Meaning
0	No action. The write to this bit is ignored.
1	It is CONSTRAINED UNPREDICTABLE whether a SW_INCR event is generated for event counter n.

## Accessing the PMSWINC\_EL0

PMSWINC\_EL0 can be accessed through the external debug interface:

Component	Offset
PMU	0xCA0

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.

# PMVIDSR, VMID Sample Register

The PMVIDSR characteristics are:

## Purpose

Contains the sampled VMID value that is captured on reading [PMPCSR](#)[31:0].

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

## Configuration

PMVIDSR 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 if the OPTIONAL PC Sample-based Profiling Extension is implemented and ARMv8.2-PCSample is implemented. If the OPTIONAL PC Sample-based Profiling Extension is implemented and ARMv8.2-PCSample is not implemented, this register is not implemented and the architecture defines the functionality in [EDVIDSR](#).

If EL2 is not implemented, this register is RES0.

This register is introduced in ARMv8.2.

## Attributes

PMVIDSR is a 32-bit register.

## Field descriptions

The PMVIDSR 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	VMID															

### Bits [31:16]

Reserved, RES0.

### VMID, bits [15:0]

VMID sample. The VMID associated with the most recent [PMPCSR](#) sample.

- If EL2 is implemented and is using AArch64, then the VMID is held in [VTTBR\\_EL2](#).VMID.
- If EL2 is implemented and is using AArch32, then the VMID is held in [VTTBR](#).VMID.
- This field is set to an UNKNOWN value if:
  - [PMPCSR](#).NS == 0.
  - [PMPCSR](#).EL == 0b10.
  - [PMPCSR](#).NS == 1, [PMPCSR](#).EL == 0b00, EL2 is using AArch64, HCR\_EL2.E2H == 1, and HCR\_EL2.TGE == 1.

- If EL2 is not implemented, then this field is RES0.
- If 16-bit VMIDs are not supported, PMVIDSR.VMID[15:8] is RES0.
- If 16-bit VMIDs are supported, but VTTBRx.VMID[15:8] are not used, PMVIDSR.VMID[15:8] is set to RES0.

When this register has an architecturally-defined reset value, this field resets to a value that is architecturally UNKNOWN.

## Accessing the PMVIDSR

PMVIDSR can be accessed through the external debug interface:

Component	Offset
PMU	0x20C

02/05/2017 15:43

Copyright © 2010-2017 ARM Limited or its affiliates. All rights reserved. This document is Non-Confidential.