



# Arm<sup>®</sup> Cortex<sup>®</sup>-X2 Core

Revision: r2p1

## Technical Reference Manual

**Non-Confidential**

**Issue 07**

Copyright © 2019–2021 Arm Limited (or its affiliates). All rights reserved. 101803\_0201\_07\_en



## Arm® Cortex®-X2 Core Technical Reference Manual

Copyright © 2019–2021 Arm Limited (or its affiliates). All rights reserved.

### Release Information

#### Document history

Issue	Date	Confidentiality	Change
0000-01	25 October 2019	Confidential	First alpha release for r0p0
0000-02	24 January 2020	Confidential	First beta release for r0p0
0000-03	30 March 2020	Confidential	First limited access release for r0p0
0100-04	12 June 2020	Confidential	First limited access release for r1p0
0200-05	21 August 2020	Confidential	First early access release for r2p0
0200-06	25 May 2021	Non-Confidential	Second early access release for r2p0
0201-07	10 December 2021	Non-Confidential	First release for r2p1

### Proprietary Notice

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

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

THIS DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, 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 click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

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

Copyright © 2019–2021 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

## Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

## Product Status

The information in this document is Final, that is for a developed product.

## Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used terms that can be offensive. Arm strives to lead the industry and create change.

This document includes terms that can be offensive. We will replace these terms in a future issue of this document.

If you find offensive terms in this document, please contact [terms@arm.com](mailto:terms@arm.com).

# Contents

<b>1 Introduction.....</b>	<b>18</b>
1.1 Product revision status.....	18
1.2 Intended audience.....	18
1.3 Conventions.....	18
1.4 Additional reading.....	20
1.5 Feedback.....	21
<b>2 The Cortex®-X2 core.....</b>	<b>22</b>
2.1 Cortex®-X2 core features.....	23
2.2 Cortex®-X2 core configuration options.....	24
2.3 DSU-110 dependent features.....	24
2.4 Supported standards and specifications.....	25
2.5 Test features.....	28
2.6 Design tasks.....	29
2.7 Product revisions.....	30
<b>3 Technical overview.....</b>	<b>31</b>
3.1 Core components.....	31
3.2 Interfaces.....	35
3.3 Programmers model.....	35
<b>4 Clocks and resets.....</b>	<b>36</b>
<b>5 Power management.....</b>	<b>37</b>
5.1 Voltage and power domains.....	37
5.2 Architectural clock gating modes.....	39
5.2.1 Wait for Interrupt and Wait for Event.....	39
5.2.2 Low-power state behavior considerations.....	40
5.3 Power control.....	41
5.4 Core power modes.....	41
5.4.1 On mode.....	43
5.4.2 Off mode.....	43
5.4.3 Emulated off mode.....	44

5.4.4 Full retention mode.....	44
5.4.5 Debug recovery mode.....	45
5.4.6 Warm reset mode.....	45
5.5 Cortex®-X2 core powerup and powerdown sequence.....	46
5.5.1 Managing RAS fault and error interrupts during the core powerdown.....	46
5.6 Debug over powerdown.....	47
<b>6 Memory management.....</b>	<b>49</b>
6.1 Memory Management Unit components.....	49
6.2 Translation Lookaside Buffer entry content.....	51
6.3 Translation Lookaside Buffer match process.....	51
6.4 Translation table walks.....	52
6.5 Hardware management of the Access flag and dirty state.....	53
6.6 Responses.....	53
6.7 Memory behavior and supported memory types.....	55
6.8 Page-based hardware attributes.....	56
<b>7 L1 instruction memory system.....</b>	<b>58</b>
7.1 L1 instruction cache behavior.....	58
7.2 L1 instruction cache Speculative memory accesses.....	59
7.3 Program flow prediction.....	59
<b>8 L1 data memory system.....</b>	<b>61</b>
8.1 L1 data cache behavior.....	61
8.2 Instruction implementation in the L1 data memory system.....	62
8.3 Internal exclusive monitor.....	63
8.4 Data prefetching.....	63
8.5 Write streaming mode.....	64
<b>9 L2 memory system.....</b>	<b>66</b>
9.1 L2 cache.....	66
9.2 Support for memory types.....	66
9.3 Transaction capabilities.....	67
<b>10 Direct access to internal memory.....</b>	<b>68</b>
10.1 L1 cache encodings.....	68
10.1.1 L1 instruction tag RAM returned data.....	71
10.1.2 L1 instruction data RAM returned data.....	71

10.1.3 L1 BTB RAM returned data.....	72
10.1.4 L1 GHB RAM returned data.....	72
10.1.5 L1 BIM RAM returned data.....	73
10.1.6 L1 instruction TLB returned data.....	73
10.1.7 L0 macro-operation RAM returned data.....	75
10.1.8 L1 data tag RAM returned data.....	76
10.1.9 L1 data data RAM returned data.....	77
10.1.10 L1 data TLB returned data.....	77
10.2 L2 cache encodings.....	79
10.2.1 L2 tag RAM returned data.....	81
10.2.2 L2 data RAM returned data.....	82
10.2.3 L2 TLB RAM returned data.....	83
10.2.4 L2 Victim RAM returned data.....	85
<b>11 RAS Extension support.....</b>	<b>86</b>
11.1 Cache protection behavior.....	86
11.2 Error containment.....	88
11.3 Fault detection and reporting.....	88
11.4 Error detection and reporting.....	88
11.4.1 Error reporting and performance monitoring.....	89
11.5 Error injection.....	89
11.6 AArch64 RAS register summary.....	90
<b>12 GIC CPU interface.....</b>	<b>91</b>
12.1 Disable the GIC CPU interface.....	91
12.2 AArch64 GIC register summary.....	92
<b>13 Advanced SIMD and floating-point support.....</b>	<b>93</b>
<b>14 Scalable Vector Extensions support.....</b>	<b>94</b>
<b>15 System control.....</b>	<b>95</b>
15.1 AArch64 Identification register summary.....	95
<b>16 Debug.....</b>	<b>97</b>
16.1 Supported debug methods.....	98
16.2 Debug register interfaces.....	99
16.2.1 Core interfaces.....	99

16.2.2 Effects of resets on debug registers.....	100
16.2.3 External access permissions to Debug registers.....	100
16.2.4 Breakpoints and watchpoints.....	101
16.3 Debug events.....	101
16.4 Debug memory map and debug signals.....	101
16.5 ROM table.....	102
16.6 CoreSight component identification.....	102
16.7 AArch64 Debug register summary.....	103
16.8 Debug register summary.....	103
16.9 CoreROM register summary.....	104
<b>17 Performance Monitors Extension support.....</b>	<b>105</b>
17.1 Performance monitors events.....	105
17.2 Performance monitors interrupts.....	115
17.3 External register access permissions.....	115
17.4 AArch64 Performance Monitors register summary.....	115
17.5 PMU register summary.....	116
<b>18 Embedded Trace Extension support.....</b>	<b>118</b>
18.1 Trace unit resources.....	119
18.2 Trace unit generation options.....	119
18.3 Reset the trace unit.....	120
18.4 Program and read the trace unit registers.....	121
18.5 Trace unit register interfaces.....	123
18.6 Interaction with the Performance Monitoring Unit and Debug.....	123
18.7 ETE events.....	124
18.8 AArch64 Trace register summary.....	124
18.9 ETE register summary.....	125
<b>19 Trace Buffer Extension support.....</b>	<b>127</b>
19.1 Program and read the trace buffer registers.....	127
19.2 Trace buffer register interface.....	127
<b>20 Activity Monitors Extension support.....</b>	<b>128</b>
20.1 Activity monitors access.....	128
20.2 Activity monitors counters.....	129
20.3 Activity monitors events.....	129



20.4 AArch64 Activity Monitors register summary.....	130
20.5 AMU register summary.....	130

## **A AArch64 system registers..... 132**

A.1 Generic system control register summary.....	132
A.1.1 AIDR_EL1, Auxiliary ID Register.....	134
A.1.2 ACTLR_EL1, Auxiliary Control Register (EL1).....	135
A.1.3 ACTLR_EL2, Auxiliary Control Register (EL2).....	136
A.1.4 HACR_EL2, Hypervisor Auxiliary Control Register.....	139
A.1.5 ACTLR_EL3, Auxiliary Control Register (EL3).....	141
A.1.6 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2).....	143
A.1.7 LORID_EL1, LORegionID (EL1).....	146
A.1.8 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register (EL1).....	147
A.1.9 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register (EL3).....	150
A.1.10 RMR_EL3, Reset Management Register (EL3).....	151
A.1.11 IMP_CPUACTLR_EL1, CPU Auxiliary Control Register (EL1).....	153
A.1.12 IMP_CPUACTLR2_EL1, CPU Auxiliary Control Register 2 (EL1).....	154
A.1.13 IMP_CPUACTLR3_EL1, CPU Auxiliary Control Register 3 (EL1).....	156
A.1.14 IMP_CPUACTLR4_EL1, CPU Auxiliary Control Register 4 (EL1).....	157
A.1.15 IMP_CPUECTLR_EL1, CPU Extended Control Register.....	159
A.1.16 IMP_CPUECTLR2_EL1, CPU Extended Control Register 2.....	168
A.1.17 IMP_CPUPPMCR3_EL3, CPU Power Performance Management Control Register.....	173
A.1.18 IMP_CPUPWRCTLR_EL1, CPU Power Control Register.....	174
A.1.19 IMP_ATCR_EL1, CPU Auxiliary Translation Control Register (EL1).....	177
A.1.20 IMP_CPUACTLR5_EL1, CPU Auxiliary Control Register 5 (EL1).....	179
A.1.21 IMP_CPUACTLR6_EL1, CPU Auxiliary Control Register 6 (EL1).....	181
A.1.22 IMP_CPUACTLR7_EL1, CPU Auxiliary Control Register 7 (EL1).....	182
A.1.23 IMP_ATCR_EL2, CPU Auxiliary Translation Control Register (EL2).....	184
A.1.24 IMP_AVTCR_EL2, CPU Virtualization Auxiliary Translation Control Register (EL2).....	186
A.1.25 IMP_CPUPPMCR_EL3, CPU Power Performance Management Control Register.....	188
A.1.26 IMP_CPUPPMCR2_EL3, CPU Power Performance Management Control Register.....	190
A.1.27 IMP_CPUPPMCR4_EL3, CPU Power Performance Management Control Register.....	191
A.1.28 IMP_CPUPPMCR5_EL3, CPU Power Performance Management Control Register.....	192
A.1.29 IMP_CPUPPMCR6_EL3, CPU Power Performance Management Control Register.....	194
A.1.30 IMP_CPUACTLR_EL3, CPU Auxiliary Control Register (EL3).....	195
A.1.31 IMP_ATCR_EL3, CPU Auxiliary Translation Control Register (EL2).....	196

A.1.32 IMP_CPUPSELR_EL3, Selected Instruction Private Select Register.....	198
A.1.33 IMP_CPUPCR_EL3, Selected Instruction Private Control Register.....	200
A.1.34 IMP_CPUPOR_EL3, Selected Instruction Private Opcode Register.....	201
A.1.35 IMP_CPUPMR_EL3, Selected Instruction Private Mask Register.....	202
A.1.36 IMP_CPUPOR2_EL3, Selected Instruction Private Opcode Register 2.....	204
A.1.37 IMP_CPUPMR2_EL3, Selected Instruction Private Mask Register 2.....	205
A.1.38 IMP_CPUPFR_EL3, Selected Instruction Private Flag Register.....	206
A.1.39 FPCR, Floating-point Control Register.....	208
A.1.40 AFSR0_EL2, Auxiliary Fault Status Register 0 (EL2).....	211
A.1.41 AFSR1_EL2, Auxiliary Fault Status Register 1 (EL2).....	214
A.1.42 AFSR0_EL1, Auxiliary Fault Status Register 0 (EL1).....	216
A.1.43 AFSR1_EL1, Auxiliary Fault Status Register 1 (EL1).....	218
A.1.44 AFSR0_EL3, Auxiliary Fault Status Register 0 (EL3).....	221
A.1.45 AFSR1_EL3, Auxiliary Fault Status Register 1 (EL3).....	222
A.2 AArch64 Debug register summary.....	224
A.2.1 IMP_IDATA0_EL3, Instruction Register 0.....	224
A.2.2 IMP_IDATA1_EL3, Instruction Register 0.....	225
A.2.3 IMP_IDATA2_EL3, Instruction Register 0.....	226
A.2.4 IMP_DDATA0_EL3, Data Register 0.....	227
A.2.5 IMP_DDATA1_EL3, Data Register 1.....	228
A.2.6 IMP_DDATA2_EL3, Data Register 2.....	229
A.3 System instruction register summary.....	230
A.3.1 SYS_IMP_RAMINDEX, RAM Index.....	230
A.4 AArch64 Identification register summary.....	232
A.4.1 MIDR_EL1, Main ID Register.....	232
A.4.2 MPIDR_EL1, Multiprocessor Affinity Register.....	234
A.4.3 REVIDR_EL1, Revision ID Register.....	236
A.4.4 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0.....	237
A.4.5 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1.....	240
A.4.6 ID_AA64ZFR0_EL1, SVE Feature ID register 0.....	241
A.4.7 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0.....	243
A.4.8 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1.....	245
A.4.9 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0.....	247
A.4.10 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1.....	248
A.4.11 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0.....	249
A.4.12 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1.....	252

A.4.13 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0.....	255
A.4.14 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1.....	257
A.4.15 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2.....	259
A.4.16 CLIDR_EL1, Cache Level ID Register.....	261
A.4.17 GMID_EL1, Multiple tag transfer ID register.....	265
A.4.18 CTR_ELO, Cache Type Register.....	266
A.4.19 DCZID_ELO, Data Cache Zero ID register.....	268
A.4.20 MPAMIDR_EL1, MPAM ID Register (EL1).....	270
A.4.21 IMP_CPUCFR_EL1, CPU Configuration Register.....	271
A.5 AArch64 Performance Monitors register summary.....	273
A.5.1 PMMIR_EL1, Performance Monitors Machine Identification Register.....	273
A.5.2 PMCR_ELO, Performance Monitors Control Register.....	275
A.5.3 PMCEID0_ELO, Performance Monitors Common Event Identification register 0.....	279
A.5.4 PMCEID1_ELO, Performance Monitors Common Event Identification register 1.....	286
A.6 AArch64 GIC register summary.....	293
A.6.1 ICC_CTLR_EL1, Interrupt Controller Control Register (EL1).....	293
A.6.2 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register.....	297
A.6.3 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Registers.....	300
A.6.4 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers.....	301
A.6.5 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Registers.....	303
A.6.6 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers.....	304
A.6.7 ICH_VTR_EL2, Interrupt Controller VGIC Type Register.....	305
A.6.8 ICC_CTLR_EL3, Interrupt Controller Control Register (EL3).....	307
A.7 AArch64 Activity Monitors register summary.....	311
A.7.1 AMEVTYPER10_ELO, Activity Monitors Event Type Registers 1.....	312
A.7.2 AMEVTYPER11_ELO, Activity Monitors Event Type Registers 1.....	313
A.7.3 AMEVTYPER12_ELO, Activity Monitors Event Type Registers 1.....	314
A.7.4 AMCFGR_ELO, Activity Monitors Configuration Register.....	316
A.7.5 AMCGCR_ELO, Activity Monitors Counter Group Configuration Register.....	318
A.7.6 AMEVTYPER00_ELO, Activity Monitors Event Type Registers 0.....	319
A.7.7 AMEVTYPER01_ELO, Activity Monitors Event Type Registers 0.....	320
A.7.8 AMEVTYPER02_ELO, Activity Monitors Event Type Registers 0.....	321
A.7.9 AMEVTYPER03_ELO, Activity Monitors Event Type Registers 0.....	322
A.8 AArch64 Trace register summary.....	323
A.8.1 TRCIDR8, ID Register 8.....	324
A.8.2 TRCIMSPECO, IMP DEF Register 0.....	325

A.8.3 TRCIDR2, ID Register 2.....	327
A.8.4 TRCIDR3, ID Register 3.....	329
A.8.5 TRCIDR4, ID Register 4.....	331
A.8.6 TRCIDR5, ID Register 5.....	333
A.8.7 TRCIDR10, ID Register 10.....	335
A.8.8 TRCIDR11, ID Register 11.....	337
A.8.9 TRCIDR12, ID Register 12.....	338
A.8.10 TRCIDR13, ID Register 13.....	339
A.8.11 TRCIDR0, ID Register 0.....	341
A.8.12 TRCIDR1, ID Register 1.....	343
A.8.13 TRCCIDCVR0, Context Identifier Comparator Value Registers <n>.....	345
A.9 MPAM register summary.....	346
A.9.1 MPAMVPMV_EL2, MPAM Virtual Partition Mapping Valid Register.....	346
A.9.2 MPAMVPM0_EL2, MPAM Virtual PARTID Mapping Register 0.....	349
A.9.3 MPAMVPM1_EL2, MPAM Virtual PARTID Mapping Register 1.....	351
A.9.4 MPAMVPM7_EL2, MPAM Virtual PARTID Mapping Register 7.....	353
A.10 AArch64 RAS register summary.....	355
A.10.1 ERRIDR_EL1, Error Record ID Register.....	355
A.10.2 ERRSELR_EL1, Error Record Select Register.....	357
A.10.3 ERXFR_EL1, Selected Error Record Feature Register.....	358
A.10.4 ERXCTLR_EL1, Selected Error Record Control Register.....	361
A.10.5 ERXSTATUS_EL1, Selected Error Record Primary Status Register.....	364
A.10.6 ERXADDR_EL1, Selected Error Record Address Register.....	369
A.10.7 ERXPFGF_EL1, Selected Pseudo-fault Generation Feature register.....	371
A.10.8 ERXPFGCTL_EL1, Selected Pseudo-fault Generation Control register.....	374
A.10.9 ERXPFGCDN_EL1, Selected Pseudo-fault Generation Countdown register.....	377
A.10.10 ERXMISCO_EL1, Selected Error Record Miscellaneous Register 0.....	378
A.10.11 ERXMISC1_EL1, Selected Error Record Miscellaneous Register 1.....	383
A.10.12 ERXMISC2_EL1, Selected Error Record Miscellaneous Register 2.....	385
A.10.13 ERXMISC3_EL1, Selected Error Record Miscellaneous Register 3.....	387
<b>B External registers.....</b>	<b>389</b>
B.1 CoreROM register summary.....	389
B.1.1 COREROM_ROMENTRY0, Core ROM table Entry 0.....	389
B.1.2 COREROM_ROMENTRY1, Core ROM table Entry 1.....	390
B.1.3 COREROM_ROMENTRY2, Core ROM table Entry 2.....	391

B.1.4 COREROM_ROMENTRY3, Core ROM table Entry 3.....	392
B.1.5 COREROM_AUTHSTATUS, Core ROM table Authentication Status Register.....	393
B.1.6 COREROM_DEVARCH, Core ROM table Device Architecture Register.....	394
B.1.7 COREROM_DEVTYPE, Core ROM table Device Type Register.....	395
B.1.8 COREROM_PIDR4, Core ROM table Peripheral Identification Register 4.....	396
B.1.9 COREROM_PIDR0, Core ROM table Peripheral Identification Register 0.....	397
B.1.10 COREROM_PIDR1, Core ROM table Peripheral Identification Register 1.....	398
B.1.11 COREROM_PIDR2, Core ROM table Peripheral Identification Register 2.....	399
B.1.12 COREROM_PIDR3, Core ROM table Peripheral Identification Register 3.....	400
B.1.13 COREROM_CIDR0, Core ROM table Component Identification Register 0.....	400
B.1.14 COREROM_CIDR1, Core ROM table Component Identification Register 1.....	401
B.1.15 COREROM_CIDR2, Core ROM table Component Identification Register 2.....	402
B.1.16 COREROM_CIDR3, Core ROM table Component Identification Register 3.....	403
B.2 PPM register summary.....	404
B.2.1 CPUPPMCR, Power Performance Management Register.....	404
B.2.2 CPUPPMCR2, Power Performance Management Register.....	405
B.2.3 CPUPPMCR3, Power Performance Management Register.....	406
B.2.4 CPUPPMCR4, Power Performance Management Register.....	406
B.2.5 CPUPPMCR5, Power Performance Management Register.....	407
B.2.6 CPUPPMCR6, Power Performance Management Register.....	408
B.3 PMU register summary.....	409
B.3.1 PMPCSSR, Snapshot Program Counter Sample Register.....	410
B.3.2 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register.....	411
B.3.3 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register.....	412
B.3.4 PMSSSR, PMU Snapshot Status Register.....	413
B.3.5 PMCCNTSR, PMU Cycle Counter Snapshot Register.....	413
B.3.6 PMEVCNTSR0, PMU Event Counter Snapshot Register.....	414
B.3.7 PMEVCNTSR1, PMU Event Counter Snapshot Register.....	415
B.3.8 PMEVCNTSR2, PMU Event Counter Snapshot Register.....	416
B.3.9 PMEVCNTSR3, PMU Event Counter Snapshot Register.....	417
B.3.10 PMEVCNTSR4, PMU Event Counter Snapshot Register.....	417
B.3.11 PMEVCNTSR5, PMU Event Counter Snapshot Register.....	418
B.3.12 PMEVCNTSR6, PMU Event Counter Snapshot Register.....	419
B.3.13 PMEVCNTSR7, PMU Event Counter Snapshot Register.....	420
B.3.14 PMEVCNTSR8, PMU Event Counter Snapshot Register.....	421
B.3.15 PMEVCNTSR9, PMU Event Counter Snapshot Register.....	421

B.3.16 PMEVCNTR10, PMU Event Counter Snapshot Register.....	422
B.3.17 PMEVCNTR11, PMU Event Counter Snapshot Register.....	423
B.3.18 PMEVCNTR12, PMU Event Counter Snapshot Register.....	424
B.3.19 PMEVCNTR13, PMU Event Counter Snapshot Register.....	425
B.3.20 PMEVCNTR14, PMU Event Counter Snapshot Register.....	425
B.3.21 PMEVCNTR15, PMU Event Counter Snapshot Register.....	426
B.3.22 PMEVCNTR16, PMU Event Counter Snapshot Register.....	427
B.3.23 PMEVCNTR17, PMU Event Counter Snapshot Register.....	428
B.3.24 PMEVCNTR18, PMU Event Counter Snapshot Register.....	429
B.3.25 PMEVCNTR19, PMU Event Counter Snapshot Register.....	429
B.3.26 PMSSCR, PMU Snapshot Capture Register.....	430
B.3.27 PMCFGR, Performance Monitors Configuration Register.....	431
B.3.28 PMCR_ELO, Performance Monitors Control Register.....	432
B.3.29 PMCEID0, Performance Monitors Common Event Identification register 0.....	434
B.3.30 PMCEID1, Performance Monitors Common Event Identification register 1.....	437
B.3.31 PMCEID2, Performance Monitors Common Event Identification register 2.....	441
B.3.32 PMCEID3, Performance Monitors Common Event Identification register 3.....	444
B.3.33 PMMIR, Performance Monitors Machine Identification Register.....	448
B.3.34 PMDEVARCH, Performance Monitors Device Architecture register.....	449
B.3.35 PMDEVID, Performance Monitors Device ID register.....	450
B.3.36 PMDEVTYPE, Performance Monitors Device Type register.....	451
B.3.37 PMPIDR4, Performance Monitors Peripheral Identification Register 4.....	451
B.3.38 PMPIDR0, Performance Monitors Peripheral Identification Register 0.....	452
B.3.39 PMPIDR1, Performance Monitors Peripheral Identification Register 1.....	453
B.3.40 PMPIDR2, Performance Monitors Peripheral Identification Register 2.....	454
B.3.41 PMPIDR3, Performance Monitors Peripheral Identification Register 3.....	455
B.3.42 PMCIDR0, Performance Monitors Component Identification Register 0.....	456
B.3.43 PMCIDR1, Performance Monitors Component Identification Register 1.....	457
B.3.44 PMCIDR2, Performance Monitors Component Identification Register 2.....	458
B.3.45 PMCIDR3, Performance Monitors Component Identification Register 3.....	459
B.4 Debug register summary.....	460
B.4.1 EDRCR, External Debug Reserve Control Register.....	460
B.4.2 EDACR, External Debug Auxiliary Control Register.....	461
B.4.3 EDPRCR, External Debug Power/Reset Control Register.....	462
B.4.4 MIDR_EL1, Main ID Register.....	463
B.4.5 EDPFR, External Debug Processor Feature Register.....	464

B.4.6 EDDFR, External Debug Feature Register.....	466
B.4.7 EDDEVARCH, External Debug Device Architecture register.....	468
B.4.8 EDDEVID2, External Debug Device ID register 2.....	469
B.4.9 EDDEVID1, External Debug Device ID register 1.....	469
B.4.10 EDDEVID, External Debug Device ID register 0.....	470
B.4.11 EDDEVTYPE, External Debug Device Type register.....	471
B.4.12 EDPIDR4, External Debug Peripheral Identification Register 4.....	472
B.4.13 EDPIDR0, External Debug Peripheral Identification Register 0.....	473
B.4.14 EDPIDR1, External Debug Peripheral Identification Register 1.....	474
B.4.15 EDPIDR2, External Debug Peripheral Identification Register 2.....	474
B.4.16 EDPIDR3, External Debug Peripheral Identification Register 3.....	475
B.4.17 EDCIDR0, External Debug Component Identification Register 0.....	476
B.4.18 EDCIDR1, External Debug Component Identification Register 1.....	477
B.4.19 EDCIDR2, External Debug Component Identification Register 2.....	478
B.4.20 EDCIDR3, External Debug Component Identification Register 3.....	478
B.5 AMU register summary.....	479
B.5.1 AMEVTYPEP00, Activity Monitors Event Type Registers 0.....	480
B.5.2 AMEVTYPEP01, Activity Monitors Event Type Registers 0.....	481
B.5.3 AMEVTYPEP02, Activity Monitors Event Type Registers 0.....	482
B.5.4 AMEVTYPEP03, Activity Monitors Event Type Registers 0.....	482
B.5.5 AMEVTYPEP10, Activity Monitors Event Type Registers 1.....	483
B.5.6 AMEVTYPEP11, Activity Monitors Event Type Registers 1.....	484
B.5.7 AMEVTYPEP12, Activity Monitors Event Type Registers 1.....	485
B.5.8 AMEVTYPEP13, Activity Monitors Event Type Registers 1.....	486
B.5.9 AMCGCR, Activity Monitors Counter Group Configuration Register.....	487
B.5.10 AMCFGR, Activity Monitors Configuration Register.....	488
B.5.11 AMIIDR, Activity Monitors Implementation Identification Register.....	490
B.5.12 AMDEVARCH, Activity Monitors Device Architecture Register.....	491
B.5.13 AMDEVTYPE, Activity Monitors Device Type Register.....	492
B.5.14 AMPIDR4, Activity Monitors Peripheral Identification Register 4.....	492
B.5.15 AMPIDR0, Activity Monitors Peripheral Identification Register 0.....	493
B.5.16 AMPIDR1, Activity Monitors Peripheral Identification Register 1.....	494
B.5.17 AMPIDR2, Activity Monitors Peripheral Identification Register 2.....	495
B.5.18 AMPIDR3, Activity Monitors Peripheral Identification Register 3.....	496
B.5.19 AMCIDR0, Activity Monitors Component Identification Register 0.....	497
B.5.20 AMCIDR1, Activity Monitors Component Identification Register 1.....	498



B.5.21 AMCIDR2, Activity Monitors Component Identification Register 2.....	499
B.5.22 AMCIDR3, Activity Monitors Component Identification Register 3.....	500
B.6 ETE register summary.....	500
B.6.1 TRCAUXCTLR, Auxillary Control Register.....	502
B.6.2 TRCIDR8, ID Register 8.....	502
B.6.3 TRCIDR9, ID Register 9.....	503
B.6.4 TRCIDR10, ID Register 10.....	504
B.6.5 TRCIDR11, ID Register 11.....	504
B.6.6 TRCIDR12, ID Register 12.....	505
B.6.7 TRCIDR13, ID Register 13.....	506
B.6.8 TRCIMSPECO, IMP DEF Register 0.....	507
B.6.9 TRCIDR0, ID Register 0.....	507
B.6.10 TRCIDR1, ID Register 1.....	509
B.6.11 TRCIDR2, ID Register 2.....	510
B.6.12 TRCIDR3, ID Register 3.....	512
B.6.13 TRCIDR4, ID Register 4.....	514
B.6.14 TRCIDR5, ID Register 5.....	515
B.6.15 TRCIDR6, ID Register 6.....	516
B.6.16 TRCIDR7, ID Register 7.....	517
B.6.17 TRCITCTRL, Integration Mode Control Register.....	518
B.6.18 TRCCLAIMSET, Claim Tag Set Register.....	519
B.6.19 TRCCLAIMCLR, Claim Tag Clear Register.....	520
B.6.20 TRCDEVARCH, Device Architecture Register.....	520
B.6.21 TRCDEVID2, Device Configuration Register 2.....	522
B.6.22 TRCDEVID1, Device Configuration Register 1.....	522
B.6.23 TRCDEVID, Device Configuration Register.....	523
B.6.24 TRCDEVTYPE, Device Type Register.....	524
B.6.25 TRCPIDR4, Peripheral Identification Register 4.....	525
B.6.26 TRCPIDR5, Peripheral Identification Register 5.....	526
B.6.27 TRCPIDR6, Peripheral Identification Register 6.....	526
B.6.28 TRCPIDR7, Peripheral Identification Register 7.....	527
B.6.29 TRCPIDR0, Peripheral Identification Register 0.....	528
B.6.30 TRCPIDR1, Peripheral Identification Register 1.....	529
B.6.31 TRCPIDR2, Peripheral Identification Register 2.....	529
B.6.32 TRCPIDR3, Peripheral Identification Register 3.....	530
B.6.33 TRCCIDR0, Component Identification Register 0.....	531



B.6.34 TRCCIDR1, Component Identification Register 1.....	532
B.6.35 TRCCIDR2, Component Identification Register 2.....	533
B.6.36 TRCCIDR3, Component Identification Register 3.....	534
<b>C Document revisions.....</b>	<b>535</b>
C.1 Revisions.....	535

# 1 Introduction

## 1.1 Product revision status

The  $r_xp_y$  identifier indicates the revision status of the product described in this manual, for example,  $r1p2$ , where:

- $r_x$**  Identifies the major revision of the product, for example,  $r1$ .
- $p_y$**  Identifies the minor revision or modification status of the product, for example,  $p2$ .

## 1.2 Intended audience

This manual is for system designers, system integrators, and programmers who are designing or programming a *System on Chip* (SoC) that uses an Arm core.

## 1.3 Conventions

The following subsections describe conventions used in Arm documents.







### Glossary

The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: [developer.arm.com/glossary](https://developer.arm.com/glossary).

### Typographic conventions

Convention	Use
<i>italic</i>	Introduces citations.
<b>bold</b>	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
monospace	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
<b>monospace bold</b>	Denotes language keywords when used outside example code.
monospace <u>underline</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: <pre>MRC p15, 0, &lt;Rd&gt;, &lt;CRn&gt;, &lt;CRm&gt;, &lt;Opcode_2&gt;</pre>

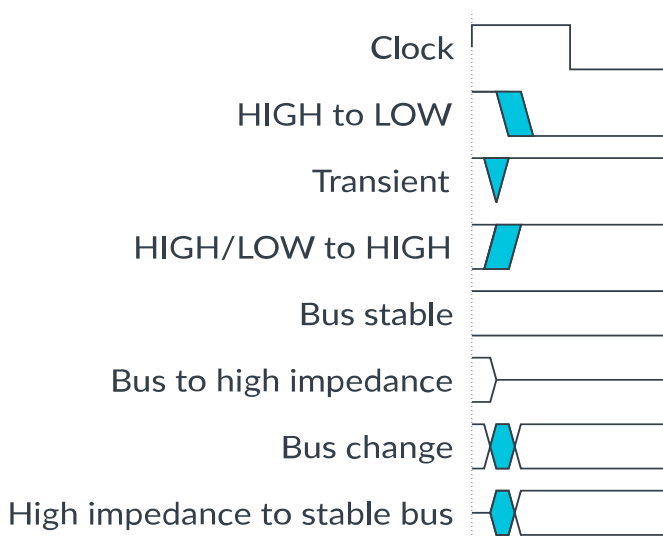
Convention	Use
<b>SMALL CAPITALS</b>	Used in body text for a few terms that have specific technical meanings, that are defined in the <i>Arm® Glossary</i> . For example, <b>IMPLEMENTATION DEFINED</b> , <b>IMPLEMENTATION SPECIFIC</b> , <b>UNKNOWN</b> , and <b>UNPREDICTABLE</b> .
 Caution	This represents a recommendation which, if not followed, might lead to system failure or damage.
 Warning	This represents a requirement for the system that, if not followed, might result in system failure or damage.
 Danger	This represents a requirement for the system that, if not followed, will result in system failure or damage.
 Note	This represents an important piece of information that needs your attention.
 Tip	This represents a useful tip that might make it easier, better or faster to perform a task.
 Remember	This is a reminder of something important that relates to the information you are reading.

## Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

**Figure 1-1: Key to timing diagram conventions**



## Signals

The signal conventions are:

### Signal level

The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

### Lowercase n

At the start or end of a signal name, n denotes an active-LOW signal.

## 1.4 Additional reading

This document contains information that is specific to this product. See the following documents for other relevant information:

**Table 1-2: Arm publications**

Document Name	Document ID	Licensee only
Cortex®-X2 Release Note	-	Yes
Arm® Cortex®-X2 Core Cryptographic Extension Technical Reference Manual	101805	No
Arm® Cortex®-X2 Core Configuration and Integration Manual	101804	Yes
Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual	101381	No
Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual	101382	Yes
Arm® Architecture Reference Manual Armv8, for A-profile architecture	DDI 0487	No
Arm® Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM) for Armv8-A	DDI 0598	No
Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile	DDI 0608	No
Arm® Reliability, Availability, and Serviceability (RAS) Specification Armv8, for the Armv8-A architecture profile	DDI 0587	No
Arm® Architecture Reference Manual Supplement The Scalable Vector Extension (SVE) for Armv8-A	DDI 0584	No
AMBA® 5 CHI Architecture Specification	IHI 0050	No

Document Name	Document ID	Licensee only
Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4	IHI 0069	No
Arm® CoreSight™ Architecture Specification v3.0	IHI 0029	No
Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual	101089	No

**Table 1-3: Other publications**

Document ID	Document Name
-	-

## 1.5 Feedback

Arm welcomes feedback on this product and its documentation.

### Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

### Feedback on content

Information about how to give feedback on the content.

If you have comments on content then send an e-mail to [errata@arm.com](mailto:errata@arm.com). Give:

- The title Arm® Cortex®-X2 Core Technical Reference Manual.
- The number 101803\_0201\_07\_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.



Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

## 2 The Cortex®-X2 core

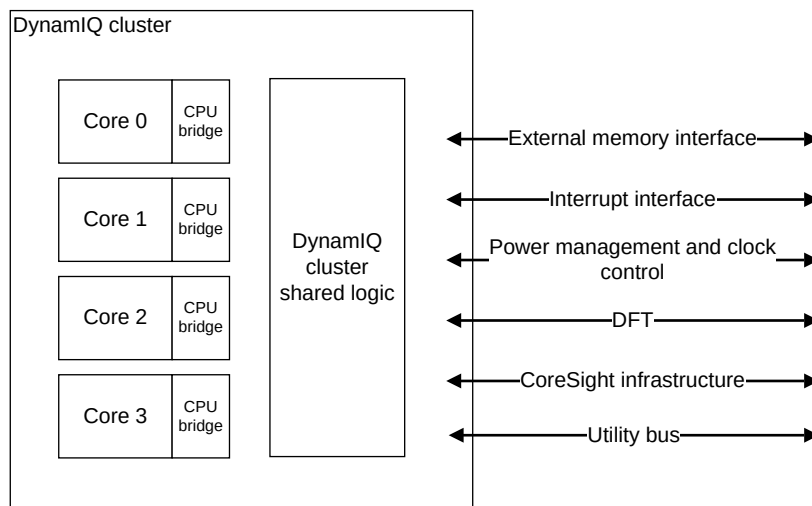
The Cortex®-X2 core is a high-performance and low-power product that implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Armv8-A architectures up to Arm®v8.5-A. The Cortex®-X2 core targets large screen compute applications.

The Cortex®-X2 core is implemented inside a DynamIQ™-110 cluster and is always connected to the *DynamIQ™ Shared Unit-110* (DSU-110) that behaves as a full interconnect with L3 cache and snoop control.

This configuration is also used in systems with different types of cores where Cortex®-X2 is the high-performance core.

The following figure shows an example configuration with four Cortex®-X2 cores in a DynamIQ™ cluster.

**Figure 2-1: Cortex®-X2 example configuration**



This manual applies to the Cortex®-X2 core only. Read this manual together with the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual* for detailed information about the DSU-110.

This manual does not provide a complete list of registers. Read this manual together with the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## 2.1 Cortex®-X2 core features

The Cortex®-X2 core might be used in standalone DynamIQ™ configurations, that is in a homogenous cluster of one to four Cortex®-X2 cores. It might also be used as the high-performance core in a heterogenous cluster.

However, regardless of the cluster configuration, the Cortex®-X2 core always has the same features.

### Core features

- Implementation of the Armv9-A A64 instruction set
- AArch64 Execution state at all Exception levels, EL0 to EL3
- *Memory Management Unit* (MMU)
- 40-bit *Physical Address* (PA) and 48-bit *Virtual Address* (VA)
- *Generic Interrupt Controller* (GIC) CPU interface to connect to an external interrupt distributor
- Generic Timers interface that supports 64-bit count input from an external system counter
- Implementation of the *Reliability, Availability, and Serviceability* (RAS) Extension
- Implementation of the *Scalable Vector Extension* (SVE) with a 128-bit vector length and *Scalable Vector Extension 2* (SVE2)
- Integrated execution unit with *Advanced Single Instruction Multiple Data* (SIMD) and floating-point support
- Support for the optional *Cryptographic Extension*, which is licensed separately
- *Activity Monitoring Unit* (AMU)

### Cache features

- Separate L1 data and instruction caches
- Private, unified data and instruction L2 cache
- Error protection on L1 instruction and data caches, L2 cache, and *MMU Translation Cache* (MMU TC) with parity or *Error Correcting Code* (ECC) allowing *Single Error Correction and Double Error Detection* (SECCDED)
- Support for *Memory System Resource Partitioning and Monitoring* (MPAM)

### Debug features

- Armv9.0-A debug logic
- *Performance Monitoring Unit* (PMU)
- *Embedded Trace Extension* (ETE)
- *Trace Buffer Extension* (TRBE)
- Optional *Embedded Logic Analyzer* (ELA)

## Related information

[3 Technical overview](#) on page 31

## 2.2 Cortex®-X2 core configuration options

You can choose the options that fit your implementation needs at build-time configuration. These options apply to all cores in the cluster.

You can configure your Cortex®-X2 core implementation using the following options:

### L2 cache size

Configure the L2 cache to be 512KB or 1MB. The cores in the cluster can have different cache sizes.

### Cryptographic Extension

Configure your implementation with or without the Cryptographic Extension. The selected option applies to all cores in the cluster.

### L2 transaction queue size

Configure the L2 transaction queue size to be 72, 80, 88, or 96.

### Coresight *Embedded Logic Analyzer* (ELA)

Include support for integrating ELA-600 as a separate licensable product.

### PMU Event Counters

Configure the number of PMU events counters to be 6 or 20.

### Size of the ATB FIFO depth in the core ELA

Configure the size of the ATB FIFO to be 4, 8, 16, 32, or 64.

### Timing closure

Configure the L2 data cache RAMs timing behavior.

See *RTL configuration process* in the *Arm® Cortex®-X2 Core Configuration and Integration Manual* for detailed configuration options and guidelines.

## 2.3 DSU-110 dependent features

Support for some *DynamiQ™ Shared Unit-110* (DSU-110) features and behaviors depends on whether your licensed core supports a particular feature.

The following table describes which DSU-110 dependent features are supported in your Cortex®-X2 core.



**Table 2-1: Cortex®-X2 core features that have a dependency on the DSU-110**

Feature	Supported in the Cortex®-X2 core	Dependency on the DSU-110
Direct connect	No	Direct connect support at the DynamIQ™-110 cluster level only applies when your licensed core also supports Direct connect.  Direct connect is intended for large systems where there are many cores.
Core included in a complex	No	Affects the DynamIQ™ cluster configuration and external signals.
Cryptographic Extension	Yes	Affects the external signals of the DSU-110.
Statistical Profiling Extension (SPE) architecture	No	



The Cryptographic Extension is supplied under a separate license.

## 2.4 Supported standards and specifications

The Cortex®-X2 core implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Armv8-A architectures up to Arm®v8.5-A. The Cortex®-X2 core also implements specific Arm architecture extensions and supports interconnect, interrupt, timer, debug, and trace architectures.

The Cortex®-X2 core supports AArch64 at all Exception levels, EL0 to EL3, and supports all mandatory features of each architecture version.

The following tables show, for each Armv8-A architecture version, the optional features that the Cortex®-X2 core supports.

**Table 2-2: Armv8.0-A optional feature support in the Cortex®-X2 core**

Feature	Status	Notes
Cryptographic Extension	Supported using a configurable option	See the <i>Arm® Cortex®-X2 Core Cryptographic Extension Technical Reference Manual</i> for more technical reference and register information. This extension is licensed separately and access to the documentation is restricted by contract with Arm.
Advanced <i>Single Instruction Multiple Data</i> (SIMD) and floating-point support	Supported	See <a href="#">13 Advanced SIMD and floating-point support</a> on page 93 for more technical reference and register information.
<i>Performance Monitoring Extension</i> (PME)	Supported	See the <i>Arm® Architecture Reference Manual Armv8, for A-profile architecture</i> for information on this feature.

**Table 2-3: Arm®v8.1-A optional feature support in the Cortex®-X2 core**

Feature	Status	Notes
FEAT_HAFDBS, Hardware management of the Access flag and dirty state	Supported	See the <i>Arm® Architecture Reference Manual Armv8, for A-profile architecture</i> for information on these features.
FEAT_VMID16, 16-bit VMID	Supported	

**Table 2-4: Arm®v8.2-A optional feature support in the Cortex®-X2 core**

Feature	Status	Notes
FEAT_HPDS2, Translation Table Page-Based Hardware Attributes	Supported	See the <i>Arm® Architecture Reference Manual Armv8, for A-profile architecture</i> for information on these features.
FEAT_PCSRv8p2, PC Sample-based profiling	Supported	
Armv8.2-SHA, FEAT_SHA512 and FEAT_SHA3 functionality	Supported as part of Armv8-A Cryptographic Extension	
FEAT_VPIPT, VMID-aware PIPT instruction cache	Not supported	
Armv8.2-SM, FEAT_SM3 and FEAT_SM4 functionality	Supported as part of Armv8-A Cryptographic Extension	
FEAT_BF16, 16-bit floating-point instructions	Supported	
FEAT_I8MM, Int8 Matrix Multiply instructions	Supported	See <a href="#">14 Scalable Vector Extensions support</a> on page 94 and the <i>Arm® Architecture Reference Manual Armv8, for A-profile architecture</i> for information on this extension.
FEAT_SVE	Supported	
FEAT_LPA, Large <i>Physical Address</i> (PA) and <i>Intermediate PA</i> (IPA) support	Not supported	-
FEAT_LVA, Large <i>Virtual Address</i> (VA) support	Not supported	-
FEAT_LSMAOC, Load/Store Multiple Atomicity and Ordering Controls	Not supported	-
FEAT_AA32HPD, AArch32 Hierarchical Permission Disables	Not supported	-
FEAT_SPE	Not supported	-

**Table 2-5: Arm®v8.3-A optional feature support in the Cortex®-X2 core**

Feature	Status	Notes
FEAT_NV, Nested Virtualization	Not supported	-
FEAT_CCIDX, Cache extended number of sets	Supported	See the <i>Arm® Architecture Reference Manual Armv8, for A-profile architecture</i> for information on this feature.
FEAT_Pauth2, Pointer Authentication enhancements	Supported	-

Feature	Status	Notes
FEAT_FPAC, <i>Faulting Pointer Authentication Code</i> (FPAC)	Supported	-

**Table 2-6: Arm®v8.4-A optional feature support in the Cortex®-X2 core**

Feature	Status	Notes
FEAT_AMUv1	Supported	See the <i>Arm® Architecture Reference Manual Armv8, for A-profile architecture</i> for information on this feature.
FEAT_MPAM	Supported	See the <i>Arm® Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM)</i> , for Armv8-A for information on this extension.
FEAT_NV2, enhanced support for Nested Virtualization	Not supported	-

**Table 2-7: Arm®v8.5-A optional feature support in the Cortex®-X2 core**

Feature	Status	Notes
FEAT_MTE, <i>Memory Tagging Extension</i> (MTE)	Supported	The Cortex®-X2 core always implements MTE and therefore is compliant with the CHI Issue E protocol.  See <i>CHI master interface</i> in the <i>Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual</i> for information on CHI.E commands inferred by MTE.
FEAT_RNG, Random Number Generator instructions	Not supported	-
FEAT_ExS, Context Synchronization and Exception Handling	Not supported	-

The following table shows the Arm®v9.0-A features that the Cortex®-X2 core supports.

**Table 2-8: Arm®v9.0-A feature support in the Cortex®-X2 core**

Feature	Status	Notes
FEAT_SVE2	Supported	See <a href="#">14 Scalable Vector Extensions support</a> on page 94.
FEAT_ETE	Supported	See <a href="#">18 Embedded Trace Extension support</a> on page 118.
FEAT_TRBE	Supported	See <a href="#">19 Trace Buffer Extension support</a> on page 127.
FEAT_SVE_SM4	Supported	These algorithms are <b>UNDEFINED</b> if the Cryptographic Extension is not enabled.
FEAT_SVE_SHA3	Supported	
FEAT_SVE_AES	Supported	
FEAT_SVE_BitPerm	Supported	-
FEAT_TME	Not supported	-

The following table shows the other standards and specifications that the Cortex®-X2 core supports.

**Table 2-9: Other standards and specifications support in the Cortex®-X2 core**

Standard or specification	Version	Notes
FEAT_GICv4p1	GICv4.1	See the <i>Arm® Generic Interrupt Controller Architecture Specification</i> , GIC architecture version 3 and version 4 for more information.
FEAT_Debugv8p4, Debug Relaxations and Extensions	-	Arm®v9.0-A architecture implemented with Arm®v8.4-A Debug architecture support and Arm®v8.3-A debug over powerdown support  See the <i>Arm®v8.5 Debug Architecture</i> for information on this architecture.
CoreSight	v3.0	See the <i>Arm® CoreSight™ Architecture Specification</i> v3.0 for more information.
FEAT_RAS	-	All extensions up to Arm®v9.0-A with <i>Error Correcting Code (ECC)</i> configured.  See <a href="#">11 RAS Extension support</a> on page 86 for more information on the implementation of this extension in the core.

### Related information

[3.1 Core components](#) on page 31

## 2.5 Test features

The Cortex®-X2 core provides test signals that enable the use of both *Automatic Test Pattern Generation (ATPG)* and *Memory Built-In Self Test (MBIST)* to test the core logic and memory arrays.

The Cortex®-X2 core includes an ATPG test interface that provides signals to control the *Design for Test (DFT)* features of the core. To prevent problems with DFT implementation, you must carefully consider how you use these signals.

Arm also provides MBIST interfaces that enable you to test the RAMs at operational frequency. You can add your own MBIST controllers to automatically generate test patterns and perform result comparisons. Optionally, you can use your EDA tool to test the physical RAMs directly instead of using the supplied Arm interfaces.

See *Design for Test integration guidelines* in the *Arm® Cortex®-X2 Core Configuration and Integration Manual* for the list of test signals and information on their usage. See also *Design for Test integration guidelines* in the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for the list of external scan control signals.



The *Arm® Cortex®-X2 Core Configuration and Integration Manual* and *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* are confidential documents that are available with the appropriate product licenses.

## 2.6 Design tasks

The Cortex®-X2 core is delivered as a synthesizable RTL description in SystemVerilog. Before you can use the Cortex®-X2 core, you must implement, integrate, and program it.

A different party can perform each of the following tasks:

### Implementation

The implementer configures the RTL, adds vendor cells/RAMs, and takes the design through the synthesis and place and route (P&R) steps to produce a hard macrocell.

The implementer chooses the options that affect how the RTL source files are rendered. These options can affect the area, maximum frequency, power, and features of the resulting macrocell.

Other components such as DFT structures and, if necessary, power switches can be added to the implementation flow.

### Integration

The integrator connects the macrocell into a SoC. This task includes connecting it to a memory system and peripherals.

The integrator configures some features of the core by tying inputs to specific values. These configuration settings affect the start-up behavior before any software configuration is made and can also limit the options available to the software.

### Software programming

The system programmer develops the software to configure and initialize the core and tests the application software.

The programmer configures the core by programming values into registers. The programmed values affect the behavior of the core.

The operation of the final device depends on the build configuration, the configuration inputs, and the software configuration.

See *RTL configuration process* in the *Arm® Cortex®-X2 Core Configuration and Integration Manual* and in the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for implementation options. See also *Functional integration* in the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for signal descriptions.

## 2.7 Product revisions

The following table indicates the main differences in functionality between product revisions.

**Table 2-10: Product revisions**

Revision	Notes
r0p0	First release
r1p0	Second release
r2p0	Third release. Added support for 20 PMU counter configuration.
r2p1	First release

Changes in functionality that have an impact on the documentation also appear in [C.1 Revisions](#) on page 535.

## 3 Technical overview

All components in the Cortex®-X2 core are always present. These components are designed to make the Cortex®-X2 core a high-performance core.

The main blocks include:

- The L1 instruction and L1 data memory systems
- The L2 memory system
- The register rename
- The instruction decode
- The instruction issue
- The execution pipeline
- The *Memory Management Unit* (MMU)
- The Trace unit and Trace buffer
- The *Performance Monitoring Unit* (PMU)
- The *Activity Monitors Unit* (AMU)
- The *Generic Interrupt Controller* (GIC) CPU interface

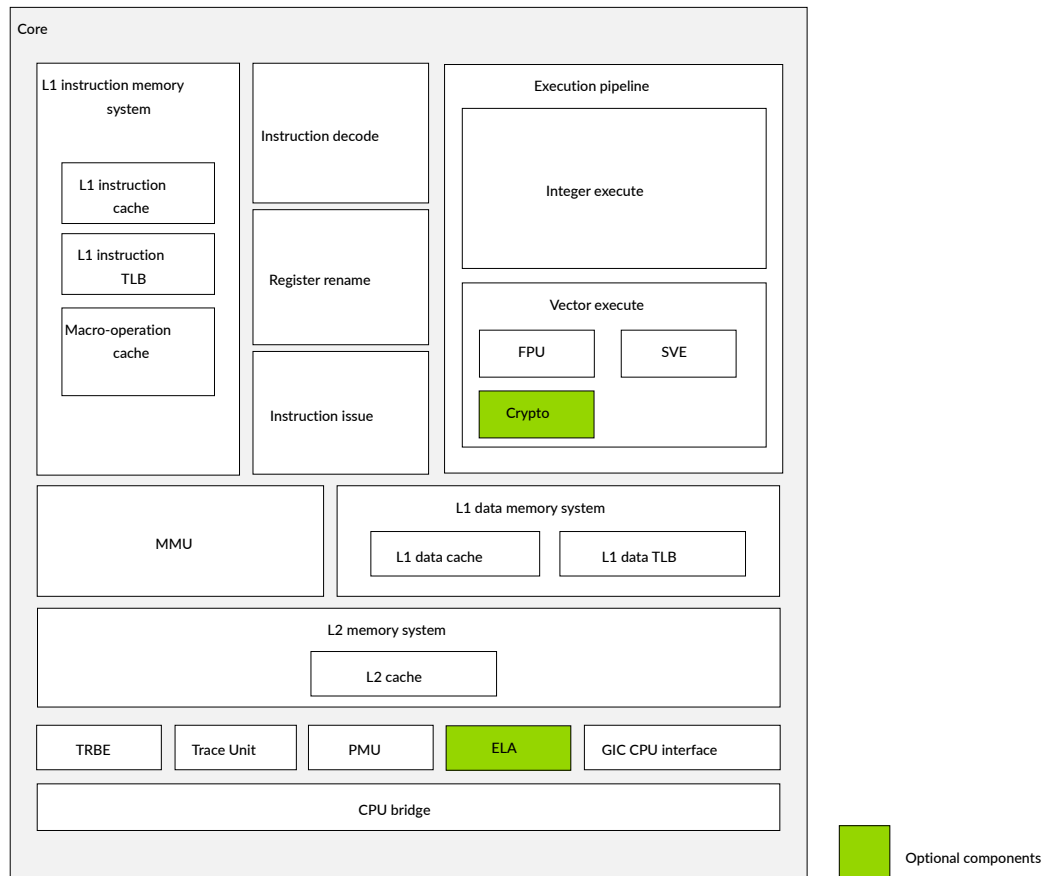
The Cortex®-X2 core interfaces with the DSU-110 through the CPU bridge.

The Cortex®-X2 core implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Armv8-A architectures up to Arm®v8.5-A. The programmers model and the architecture features implemented, such as the Generic Timer, are compliant with the standards in [2.4 Supported standards and specifications](#) on page 25.

### 3.1 Core components

The Cortex®-X2 core includes components designed to make it a high-performance and low-power product. The Cortex®-X2 core includes a CPU bridge that connects the core to the *DynamiQ™ Shared Unit-110* (DSU-110). The DSU-110 connects the core to an external memory system and the rest of the *System on Chip* (SoC).

The following figure shows the Cortex®-X2 core components.

**Figure 3-1: Cortex®-X2 core components**

## L1 instruction memory system

The L1 instruction memory system fetches instructions from the instruction cache and delivers the instruction stream to the instruction decode unit.

The L1 instruction memory system includes:

- A 64KB, 4-way set associative L1 instruction cache with 64-byte cache lines.
- A fully associative L1 instruction *Translation Lookaside Buffer* (TLB) with native support for 4KB, 16KB, 64KB, and 2MB page sizes.
- A 3072-entry, 4-way skewed associative *L0 Macro-OP* (MOP) cache, which contains decoded and optimized instructions for higher performance.
- A dynamic branch predictor.

## Instruction decode

The instruction decode unit decodes AArch64 instructions into internal format.



## Register rename

The register rename unit performs register renaming to facilitate out-of-order execution and dispatches decoded instructions to various issue queues.

## Instruction issue

The instruction issue unit controls when the decoded instructions are dispatched to the execution pipelines. It includes issue queues for storing instructions pending dispatch to execution pipelines.

## Integer execute

The integer execution pipeline is part of the overall execution pipeline and includes the integer execute unit that performs arithmetic and logical data processing operations.

## Vector execute

The vector execute unit is part of the execution pipeline and performs Advanced SIMD and floating-point operations (FPU), executes the *Scalable Vector Extension* (SVE) and *Scalable Vector Extension 2* (SVE2) instructions, and can optionally execute the cryptographic instructions (Crypto).

## Advanced SIMD and floating-point support

Advanced SIMD is a media and signal processing architecture that adds instructions primarily for audio, video, 3D graphics, image, and speech processing. The floating-point architecture provides support for single-precision and double-precision floating-point operations.

## Cryptographic Extension

The Cryptographic Extension is optional in Cortex®-X2 cores. The Cryptographic Extension adds new instructions to the Advanced SIMD and the *Scalable Vector Extension* (SVE) instruction sets that accelerate:

- *Advanced Encryption Standard* (AES) encryption and decryption.
- The *Secure Hash Algorithm* (SHA) functions SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512.
- Armv8.2-SM SM3 hash function and SM4 encryption and decryption instructions.
- Finite field arithmetic that is used in algorithms such as Galois/Counter Mode and Elliptic Curve Cryptography.



Note

The optional Cryptographic Extension is not included in the base product. Arm supplies the Cryptographic Extension only under an additional license to the Cortex®-X2 core license.

---

## Scalable Vector Extension

The *Scalable Vector Extension* (SVE) is an extension to the Armv8-A architecture.

It complements but does not replace AArch64 Advanced SIMD and floating-point functionality.



The Advanced SIMD architecture, its associated implementations, and supporting software, are also referred to as NEON™ technology.

## L1 data memory system

The L1 data memory system executes load and store instructions and encompasses the L1 data side memory system. It also services memory coherency requests.

The L1 data memory system includes:

- A 64KB, 4-way set associative cache with 64-byte cache lines.
- A fully associative L1 data TLB with native support for 4KB, 16KB and 64KB page sizes and 2MB and 512MB block sizes.

## Memory Management Unit

The *Memory Management Unit* (MMU) provides fine-grained memory system control through a set of virtual-to-physical address mappings and memory attributes that are held in translation tables.

These are saved into the TLB when an address is translated. The TLB entries include global and *Address Space Identifiers* (ASIDs) to prevent context switch TLB invalidations. They also include *Virtual Machine Identifiers* (VMIDs) to prevent TLB invalidations on virtual machine switches by the hypervisor.

## L2 memory system

The L2 memory system includes the L2 cache. The L2 cache is private to the core and is 8-way set associative. You can configure its RAM size to be 512KB or 1MB. The L2 memory system is connected to the DSU-110 through an asynchronous CPU bridge.

## Embedded Trace Extension and Trace Buffer Extension

The Cortex®-X2 core supports a range of debug, test, and trace options including a trace unit and trace buffer.

The Cortex®-X2 core also includes a ROM table that contains a list of components in the system. Debuggers can use the ROM table to determine which CoreSight components are implemented.

All the debug and trace components of the Cortex®-X2 core are described in this manual. The *Arm® Cortex®-X2 Core Configuration and Integration Manual* provides information about the *Embedded Logic Analyzer* (ELA).

## Performance Monitoring Unit

The *Performance Monitoring Unit* (PMU) provides six or 20, depending on your configuration, performance monitors that can be configured to gather statistics on the operation of each core and the memory system. The information can be used for debug and code profiling.

## GIC CPU interface

The GIC CPU interface, when integrated with an external distributor component, is a resource for supporting and managing interrupts in a cluster system.

## CPU bridge

In a cluster, there is one CPU bridge between each Cortex®-X2 core and the DSU-110.

The CPU bridge controls buffering and synchronization between the core and the DSU-110.

The CPU bridge is asynchronous to allow different frequency, power, and area implementation points for each core. You can configure the CPU bridge to run synchronously without affecting the other interfaces such as debug and trace which are always asynchronous.

## Related information

[6 Memory management](#) on page 49

[7 L1 instruction memory system](#) on page 58

[8 L1 data memory system](#) on page 61

[9 L2 memory system](#) on page 66

[12 GIC CPU interface](#) on page 91

[17 Performance Monitors Extension support](#) on page 105

[18 Embedded Trace Extension support](#) on page 118

## 3.2 Interfaces

The DSU-110 manages all Cortex®-X2 core external interfaces to the *System on Chip* (SoC).

See *Technical overview* in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual* for detailed information on these interfaces.

## 3.3 Programmers model

The Cortex®-X2 core implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Armv8-A architectures up to Arm®v8.5-A. The Cortex®-X2 core supports the AArch64 Execution state at all Exception levels, EL0 to EL3.

See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information about the programmers model.

## Related information

[2.4 Supported standards and specifications](#) on page 25

## 4 Clocks and resets

To provide dynamic power savings, the Cortex®-X2 core supports hierarchical clock gating. It also supports Warm and Cold resets.

Each Cortex®-X2 core has a single clock domain and receives a single clock input. This clock input is gated by an architectural clock gate in the CPU bridge.

In addition, the Cortex®-X2 core implements extensive clock gating that includes:

- Regional clock gates to various blocks that can gate off portions of the clock tree
- Local clock gates that can gate off individual registers or banks of registers

The Cortex®-X2 core receives the following reset signals from the *DynamlQ™ Shared Unit-110* (DSU-110) side of the CPU bridge:

- A Warm reset for all registers in the core except for:
  - Some parts of Debug logic
  - Some parts of trace unit logic
  - *Reliability, Availability, and Serviceability* (RAS) logic
- A Cold reset for the logic in the core, including the debug logic, trace logic, and RAS logic.

For a complete description of the clock gating and reset scheme of the core, see the following sections in the *Arm® DynamlQ™ Shared Unit-110 Technical Reference Manual*:

- *Clocks and resets*
- *Power and reset control with Power Policy Units*

## 5 Power management

The Cortex®-X2 core provides mechanisms to control both dynamic and static power dissipation.

The dynamic power management includes the following features:

- Hierarchical clock gating
- Per-core *Dynamic Voltage and Frequency Scaling* (DVFS)

The static power management includes the following features:

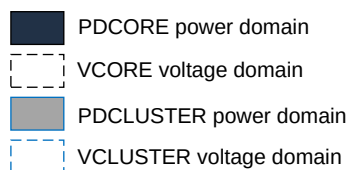
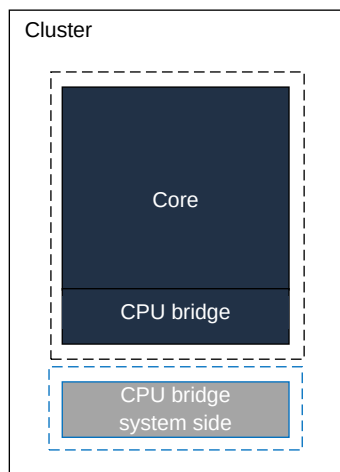
- Powerdown
- Dynamic retention, a low-power mode that retains the register and RAM state

### 5.1 Voltage and power domains

The *DynamiQ™ Shared Unit-110* (DSU-110) *Power Policy Units* (PPUs) control power management for the Cortex®-X2 core. The core supports one power domain, PDCORE, and one system power domain, PDCLUSTER. Similarly, it supports one core voltage domain, VCORE, and one cluster system voltage domain, VCLUSTER. The power domains and voltage domains have the same boundaries.

The PDCORE power domain contains all Cortex®-X2 core logic and part of the core asynchronous bridge that belongs to the VCORE domain. The PDCLUSTER power domain contains the part of the CPU bridge that belongs to the VCLUSTER domain.

The following figure shows the Cortex®-X2 core power domain and voltage domain. It also shows the cluster power domain and voltage domain that cover the system side of the CPU bridge.

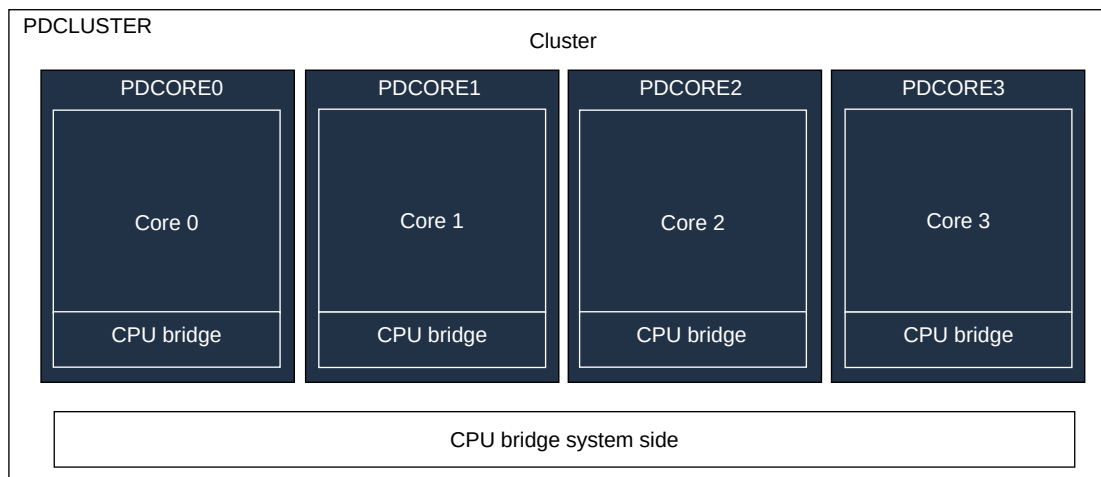
**Figure 5-1: Cortex®-X2 core voltage domains and power domains**

You can tie the VCORE and VCLUSTER voltage domains to the same supply if either:

- The core is configured to run synchronously with the DSU-110 sharing the same clock.
- The core is not required to support *Dynamic Voltage and Frequency Scaling* (DVFS).

In a cluster with multiple Cortex®-X2 cores, there is one PDCORE<n> power domain per core, where n is the core instance number. If a core is not present, then the corresponding power domain is not present.

This diagram shows the power domains for an example Cortex®-X2 configuration with a four-core cluster:

**Figure 5-2: Core power domains in a cluster with four Cortex®-X2 cores**

Clamping cells between power domains are inferred through power intent files (UPF) rather than instantiated in the RTL. See *Power management* in the *Arm® Cortex®-X2 Core Configuration and Integration Manual* for more information.



The *Arm® Cortex®-X2 Core Configuration and Integration Manual* is a confidential document that is available with the appropriate product license.

For detailed information on the DSU-110 cluster power domains and voltage domains, see *Power management* in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*.

## 5.2 Architectural clock gating modes

The `WFI` and `WFE` instructions put the core into a low-power mode. These instructions disable the clock at the top of the clock tree. The core remains fully powered and retains the state.

### 5.2.1 Wait for Interrupt and Wait for Event

*Wait for Interrupt* (WFI) and *Wait for Event* (WFE) are features that put the core in a low-power state by disabling most of the core clocks, while keeping the core powered up. When the core is in WFI or WFE state, the input clock is gated externally to the core at the CPU bridge.

There is a small amount of dynamic power used by the logic that is required to wake up the core from WFI or WFE low-power state. Other than this power use, the power that is drawn is reduced to static leakage current only.

When the core executes the `WFI` or `WFE` instruction, it waits for all instructions in the core, including explicit memory accesses, to retire before it enters a low-power state. The `WFI` and `WFE` instructions also ensure that store instructions have updated the cache or have been issued to the L3 memory system.



Executing the `WFE` instruction when the event register is set does not cause entry into low-power state, but clears the event register.

The core exits the `WFI` or `WFE` state when one of the following events occurs:

- The core detects a reset.
- The core detects one of the architecturally defined `WFI` or `WFE` wakeup events.

`WFI` and `WFE` wakeup events can include physical and virtual interrupts.

See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information about entering low-power state and wakeup events.

## 5.2.2 Low-power state behavior considerations

You must consider how certain events affect the *Wait for Interrupt* (`WFI`) and *Wait for Event* (`WFE`) low-power state behavior of the Cortex®-X2 core.

While the core is in `WFI` or `WFE` state, the clocks in the core are temporarily enabled when any of the following events are detected:

- A system snoop request that must be serviced by the core L1 data cache or the L2 cache
- A cache or *Translation Lookaside Buffer* (TLB) maintenance operation that must be serviced by the core L1 instruction cache, L1 data cache, L2 cache, or TLB
- An access on the Utility bus interface
- A *Generic Interrupt Controller* (GIC) CPU access or debug access through the *Advanced Peripheral Bus* (APB) interface



The core does not exit `WFI` or `WFE` state when the clocks are temporarily enabled.

See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information about `WFI` and `WFE`.



## 5.3 Power control

The DSU-110 *Power Policy Units* (PPUs) control all core and cluster power mode transitions.

Each of the cores have their own individual PPU for controlling their respective core power domain. For example there is a PPU for PDCORE0 and a PPU for PDCORE1.

In addition, there is a PPU for the cluster.

The PPUs decide and request any change in power mode. The Cortex®-X2 core then performs any actions necessary to reach the requested power mode. For example, the core might gate clocks, clean caches, or disable coherency before accepting the request.

See *Power management* and *Power and reset control with Power Policy Units* in the Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual for more information about the PPUs for the cluster and the cores.

### Related information

[A.1.25 IMP\\_CPUPPMCR\\_EL3, CPU Power Performance Management Control Register](#) on page 188

[B.2.1 CPUPPMCR, Power Performance Management Register](#) on page 404

## 5.4 Core power modes

The Cortex®-X2 core power domain has a defined set of power modes and corresponding legal transitions between these modes. The power mode of each core can be independent of other cores in a cluster.

The *Power Policy Unit* (PPU) of a core manages at the cluster level the transitions between the power modes for that core. See *Power management* in the Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual for more information.

The following table shows the supported Cortex®-X2 core power modes.



Caution

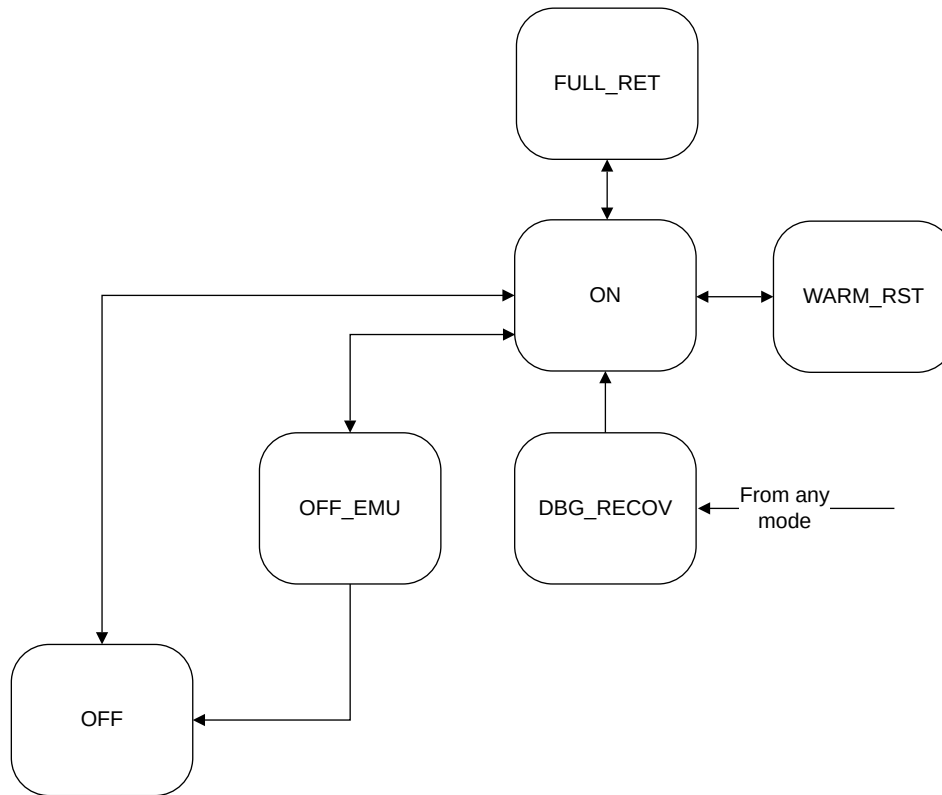
Power modes that are not shown in the following table are not supported and must not occur. Deviating from the legal power modes can lead to **UNPREDICTABLE** results. You must comply with the dynamic power management and powerup and powerdown sequences described in [5.5 Cortex-X2 core powerup and powerdown sequence](#) on page 45.

**Table 5-1: Cortex®-X2 core power modes**

Power mode	Short name	Power state
On	ON	The core is powered up and active.

Power mode	Short name	Power state
Full retention	FULL_RET	<p>The core is in retention.</p> <p>In this mode, only power that is required to retain register and RAM state is available. The core is not operational.</p> <p>A core must be in <i>Wait for Interrupt</i> (WFI) or <i>Wait for Event</i> (WFE) low-power state before it enters this mode.</p>
Off	OFF	The core is powered down.
Emulated Off	OFF_EMU	<p>Emulated off mode permits you to debug the powerup and powerdown cycle without changing the software.</p> <p>In this mode, the core powerdown is normal, except:</p> <ul style="list-style-type: none"> <li>The clock is not gated and power is not removed when the core is powered down.</li> <li>Only a Warm reset is asserted. The debug logic is preserved in the core and remains accessible by the debugger.</li> </ul>
Debug recovery	DBG_RECOV	<p>The RAM and logic are powered up.</p> <p>This mode is for applying a Warm reset to the cluster, while preserving memory and RAS registers for debug purposes. Both cache and RAS state are preserved when transitioning from DBG_RECOV to ON.</p> <p><b>Caution:</b> This mode must not be used during normal system operation.</p>
Warm reset	WARM_RST	A Warm reset resets all state except for the trace logic and the debug and RAS registers.

The following figure shows the supported modes for the Cortex®-X2 core power domain and the legal transitions between them.

**Figure 5-3: Cortex®-X2 core power mode transitions****Related information**

[5.2 Architectural clock gating modes](#) on page 39

[5.4.4 Full retention mode](#) on page 44

[5.2.1 Wait for Interrupt and Wait for Event](#) on page 39

**5.4.1 On mode**

In the On power mode, the Cortex®-X2 core is on and fully operational.

The core can be initialized into the On mode. When a transition to the On mode is completed, all caches are accessible and coherent. Other than the normal architectural steps to enable caches, no additional software configuration is required.

**5.4.2 Off mode**

In the Off power mode, power is removed completely from the core and no state is retained.

In Off mode, all core logic and RAMs are off. The domain is inoperable and all core state is lost. The L1 and L2 caches are disabled, cleaned and invalidated, and the core is removed from coherency automatically on transition to Off mode.

A Cold reset can reset the core in this mode.

An attempted debug access when the core domain is off returns an error response on the internal debug interface, indicating that the core is not available.

### 5.4.3 Emulated off mode

In Emulated off mode, all core domain logic and RAMs are kept on. All Debug registers must retain their state and be accessible from the external debug interface. All other functional interfaces behave as if the core were in Off mode.

### 5.4.4 Full retention mode

Full retention mode is a dynamic retention mode that is controlled using the *Power Policy Unit* (PPU). On wakeup, full power to the core can be restored and execution can continue.

In Full retention mode, only power that is required to retain register and RAM state is available. The core is in retention state and is non-operational.

The core enters Full retention mode when all of the following conditions are met:

- The retention timer has expired. For more information on setting the retention timer, see [A.1.18 IMP\\_CPUPWRCTLR\\_EL1, CPU Power Control Register](#) on page 174.
- The core is in *Wait for Interrupt* (WFI) or *Wait for Event* (WFE) low-power state.
- The core clock is not temporarily enabled for any of the following reasons:
  - L1 snoops or L2 snoops
  - Cache or *Translation Lookaside Buffer* (TLB) maintenance operations
  - Debug or *Generic Interrupt Controller* (GIC) access

The core exits Full retention mode when it detects any of the following events:

- A WFI or WFE wakeup event, as defined in the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.
- An event that requires the core clock to be temporarily enabled without exiting the WFI or WFE low-power state. For example:
  - L1 snoops or L2 snoops
  - Cache or TLB maintenance operations
  - Debug access from the DebugBlock of the *DynamiQ™ Shared Unit-110* (DSU)
  - GIC access

### 5.4.5 Debug recovery mode

Debug recovery mode supports debug of external watchdog-triggered reset events, such as watchdog timeout.

By default, the core invalidates its caches when it transitions from Off to On mode. Using Debug recovery mode allows the L1 cache and L2 cache contents that were present before the reset to be observable after the reset. The contents of the caches are retained and are not altered on the transition back to the On mode.

In addition to preserving the cache contents, Debug recovery supports preserving the *Reliability, Availability, and Serviceability* (RAS) state. When in Debug recovery mode, a DynamiQ™-110 cluster-wide Warm reset must be applied externally. The RAS and cache state are preserved when the core is transitioned to the On mode.



Debug recovery is strictly for debug purposes. It must not be used for functional purposes, because correct operation of the caches is not guaranteed when entering this mode.

---

Debug recovery mode can occur at any time with no guarantee of the state of the core. A request of this type is accepted immediately, therefore its effects on the core, the DynamiQ™ cluster, or the wider system are **UNPREDICTABLE**, and a wider system reset might be required. In particular, any outstanding memory system transactions at the time of the reset might complete after the reset. The core is not expecting these transactions to complete after a reset, and might cause a system deadlock.

If the system sends a snoop to the DynamiQ™ cluster during Debug recovery mode, depending on the cluster state:

- The snoop might get a response and disturb the contents of the caches
- The snoop might not get a response and cause a system deadlock

### 5.4.6 Warm reset mode

A Warm reset resets all state except for the trace logic, debug registers, and *Reliability, Availability, and Serviceability* (RAS) registers.

A Warm reset is applied to the Cortex®-X2 core when the core receives a Warm reset signal from the *DynamiQ™ Shared Unit-110* (DSU-110) side of the CPU bridge.

The Cortex®-X2 core implements the Arm®v8-A Reset Management Register, RMR\_EL3. When the core runs in EL3, it requests a Warm reset if you set the RMR\_EL3.RR bit to 1.

See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information about RMR\_EL3.

## 5.5 Cortex®-X2 core powerup and powerdown sequence

There is no specific sequence to power up the Cortex®-X2 core. To power down the core, you must follow a specific sequence. There are no software steps required to bring a core into coherence after reset.

To powerdown the Cortex®-X2 core:

1. If required, save the state of the core to system memory to allow for retrieval of the core state during core powerup.
2. Disable interrupts to the core.
  - a. Disable the interrupt enable bits in the ICC\_IGRPEN0\_EL1 and ICC\_IGRPEN1\_EL1 registers.
  - b. Set the GIC distributor wake-up request for the core using the GICR\_WAKER register.
  - c. Read the GICR\_WAKER register to confirm that the ChildrenAsleep bit indicates that the interface is quiescent.
3. Disable the interrupt outputs from the RAS registers. Alternatively, re-direct the core RAS fault and error interrupt outputs to the system error manager. For more information, see [5.5.1 Managing RAS fault and error interrupts during the core powerdown](#) on page 46.
4. Set the IMP\_CPUPWRCTLR\_EL1.CORE\_PWRDN\_EN bit to 1 to indicate to the power controller that a powerdown is requested.
5. Execute an `ISB` instruction.
6. Execute a `WFI` instruction. Once the `WFI` instruction is executed, the powerdown sequence cannot be interrupted.

After you have executed the `WFI` instruction, and subsequently received a powerdown request from the power controller, the hardware:

- Disables and cleans the core caches
- Removes the core from system coherency

When the IMP\_CPUPWRCTLR\_EL1.CORE\_PWRDN\_EN bit is set, executing a `WFI` instruction automatically masks all interrupts and wakeup events in the core. As a result, applying a reset is the only way to wake up the core from the *Wait for Interrupt* (WFI) state.

### 5.5.1 Managing RAS fault and error interrupts during the core powerdown

After the `WFI` instruction is executed, the power management architecture does not permit interrupting the core software.

Therefore, the core software cannot be interrupted to manage any RAS fault or error when either of the following is true:

- A RAS fault or error is detected before the core powerdown procedure executes the `WFI` instruction and the error has not been cleared.

- A RAS fault or error is detected after the core powerdown procedure executes the WFI instruction.

You must manage the status of the RAS fault and error interrupts to complete the core powerdown sequence. Any active RAS fault or error interrupt output from the core prevents the core from powering down, so that:

- The core is left powered ON, but the software remains inactive.
- All requests from the core PPU to power off the core are denied.
- A full cluster reset is the only mechanism available to restart the core software.

If the RAS fault and error interrupt outputs are disabled before the core powerdown procedure, and if the error detection and correction response is enabled, then the following is true:

- Correctable errors are corrected
- Deferrable errors are deferred as part of the automatic cache clean and invalidation procedures
- Error records for the correctable and deferrable errors are lost when the core is powered OFF
- If there is an uncorrectable error when the core is powering off, this error is not signaled to the system and might corrupt the system behavior

If preferable, you can disable the generation of RAS faults and error interrupts for correctable and deferrable errors while enabling the error interrupt for uncorrectable errors. However, the core error interrupt output must be re-routed to the system error manager before executing the WFI instruction in the core powerdown procedure. To do this, configure the ERxCTLR\_EL1 register as follows:

- ERxCTLR\_EL1.CFI = 0
- ERxCTLR\_EL1.FI = 0
- ERxCTLR\_EL1.UI = 1

If an uncorrectable error occurs during the powerdown, the core remains powered ON and the software remains inactive. The system error manager is then responsible for resetting the entire cluster and the wider system that interacts with the core and cluster. To use this approach, the system must be designed to allow the core RAS error interrupt to re-route to the system error manager. As the core RAS registers are only accessible to software running on the core, the system error manager is unable to identify where the uncorrectable error occurred within the core.

## 5.6 Debug over powerdown

The Cortex®-X2 core supports debug over powerdown, which allows a debugger to retain its connection with the core even when powered down. This behavior enables debug to continue

through powerdown scenarios, rather than having to re-establish a connection each time the core is powered up.

The debug over powerdown logic is part of the DebugBlock in the *DynamlQ™ Shared Unit-110* (DSU-110). The DebugBlock is external to the *DynamlQ™-110* cluster and must remain powered on during the debug over powerdown process.

See *Debug* in the *Arm® DynamlQ™ Shared Unit-110 Technical Reference Manual* for more information.



## 6 Memory management

The *Memory Management Unit* (MMU) translates an input address to an output address.

This translation is based on address mapping and memory attribute information that is available in the Cortex®-X2 core internal registers and translation tables. The MMU also controls memory access permissions, memory ordering, and cache policies for each region of memory.

An address translation from an input address to an output address is described as a stage of address translation. The Cortex®-X2 core can perform:

- Stage 1 translations that translate an input *Virtual Address* (VA) to an output *Physical Address* (PA) or *Intermediate Physical Address* (IPA).
- Stage 2 translations that translate an input IPA to an output PA.
- Combined stage 1 and stage 2 translations that translate an input VA to an IPA, and then translate that IPA to an output PA. The Cortex®-X2 core performs translation table walks for each stage of the translation.

In addition to translating an input address to an output address, a stage of address translation also defines the memory attributes of the output address. With a two-stage translation, the stage 2 translation can modify the attributes that the stage 1 translation defines. A stage of address translation can be disabled or bypassed, and cores can define memory attributes for disabled and bypassed stages of translation.

Each stage of address translation uses address translations and associated memory properties that are held in memory-mapped translation tables. Translation table entries can be cached into a *Translation Lookaside Buffer* (TLB). The translation table entries enable the MMU to provide fine-grained memory system control and to control the table walk hardware.

See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information on this feature.

### 6.1 Memory Management Unit components

The Cortex®-X2 *Memory Management Unit* (MMU) includes several *Translation Lookaside Buffers* (TLBs), an *MMU Translation Cache* (MMUTC), and a translation table prefetcher.

A TLB is a cache of recently executed page translations within the MMU. The Cortex®-X2 core implements a two-level TLB structure. The TLB stores all translation table sizes and is responsible for breaking these down into smaller tables when required for the L1 data or instruction TLB.

The following table describes the MMU components.

**Table 6-1: MMU components**

Component	Description
L1 instruction TLB	<ul style="list-style-type: none"> <li>Caches entries at the 4KB, 16KB, 64KB, or 2MB granularity of <i>Virtual Address</i> (VA) to <i>Physical Address</i> (PA) mapping only</li> <li>Fully associative</li> <li>48 entries</li> </ul>
L1 data TLB	<ul style="list-style-type: none"> <li>Caches entries at the 4KB, 16KB, 64KB, 2MB, or 512MB granularity of VA to PA mappings only</li> <li>Fully associative</li> <li>48 entries</li> </ul>
L1 <i>Trace Buffer Extension</i> (TRBE) TLB	<ul style="list-style-type: none"> <li>VA to PA translations of any page and block size</li> <li>2 entries</li> </ul>
L2 TLB	<ul style="list-style-type: none"> <li>Shared by instructions and data</li> <li>VA to PA mappings for 4KB, 16KB, 64KB, 2MB, 32MB, 512MB, and 1GB block sizes</li> <li><i>Intermediate Physical Address</i> (IPA) to PA mappings for: <ul style="list-style-type: none"> <li>2MB and 1GB block sizes in a 4KB translation granule</li> <li>32MB block size in a 16KB translation granule</li> <li>512MB block size in a 64KB granule</li> </ul> </li> <li><i>Intermediate PAs</i> (IPAs) obtained during a translation table walk</li> <li>8-way set associative</li> <li>2048 entries</li> </ul>
Translation table prefetcher	<ul style="list-style-type: none"> <li>Detects access to contiguous translation tables and prefetches the next one</li> <li>Can be disabled in the ECTLR register</li> </ul>

TLB entries contain a global indicator and an *Address Space Identifier* (ASID) to allow context switches without requiring the TLB to be invalidated.

TLB entries contain a *Virtual Machine Identifier* (VMID) to allow virtual machine switches by the hypervisor without requiring the TLB to be invalidated.

A hit in the L1 instruction TLB provides a single **CLK** cycle access to the translation, and returns the PA to the instruction cache for comparison. It also checks the access permissions to signal an Instruction Abort.

A hit in the L1 data TLB provides a single **CLK** cycle access to the translation, and returns the PA to the data cache for comparison. It also checks the access permissions to signal a Data Abort.

A miss in the L1 data TLB or a hit in the L2 TLB has a 3-cycle penalty compared to a hit in the L1 data TLB. This penalty can be increased depending on the arbitration of pending requests.

## 6.2 Translation Lookaside Buffer entry content

*Translation Lookaside Buffer* (TLB) entries store the context information required to facilitate a match and avoid the need for a TLB clean on a context or virtual machine switch.

Each TLB entry contains:

- A *Virtual Address* (VA)
- A *Physical Address* (PA)
- A set of memory properties that includes type and access permissions

Each TLB entry is associated with either:

- A particular *Address Space Identifier* (ASID)
- A global indicator

Each TLB entry also contains a field to store the *Virtual Machine Identifier* (VMID) in the entry applicable to accesses from EL0 and EL1. The VMID permits hypervisor virtual machine switches without requiring the TLB to be invalidated.

### Related information

[6.4 Translation table walks](#) on page 52

## 6.3 Translation Lookaside Buffer match process

The Armv8-A architecture provides support for multiple *Virtual Address* (VA) spaces that are translated differently.

Each TLB entry is associated with a particular translation regime.

- EL3 in Secure state
- EL2 (or EL0 in VHE mode) in Secure state
- EL1 or EL0 in Secure state
- EL1 or EL0 in Non-secure state

A TLB match entry occurs when the following conditions are met:

- The VA bits[48:N], where N is  $\log_2$  of the block size for that translation that is stored in the TLB entry, matches the requested address.
- Entry translation regime matches the current translation regime.
- The ASID matches the current ASID held in the TTBR0\_ELx or TTBR1\_ELx register associated with the target translation regime, or the entry is marked global.
- The VMID matches the current VMID held in the VTTBR\_EL2 register.

The ASID and VMID matches are ignored when ASID and VMID are not relevant. ASID is relevant when the translation regime is:

- EL2 in Secure state with HCR\_EL2.E2H and HCR\_EL2.TGE set to 1
- EL1 or EL0 in Secure state
- EL1 or EL0 in Non-secure state

VMID is relevant for EL1 or EL0 in Non-secure state when HCR\_EL2.E2H and HCR\_EL2.TGE are not both set. It is also relevant in Secure state when SCR\_EL3.EEL2 is 1.

## 6.4 Translation table walks

When the Cortex®-X2 core generates a memory access, the *Memory Management Unit* (MMU) searches for the requested *Virtual Address* (VA) in the *Translation Lookaside Buffers* (TLBs). If it is not present, then it is a miss and the MMU proceeds by looking up the translation table during a translation table walk.

When the Cortex®-X2 core generates a memory access, the MMU:

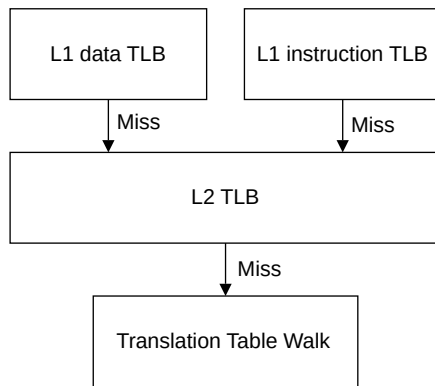
1. Performs a lookup for the requested VA, current *Address Space Identifier* (ASID), current *Virtual Machine Identifier* (VMID), and current translation regime in the relevant instruction or data L1 TLB.
2. If there is a miss in the relevant L1 TLB, the MMU performs a lookup in the L2 TLB for the requested VA, current ASID, current VMID, and translation regime.
3. If there is a miss in the L2 TLB, the MMU performs a hardware translation table walk.

Address translation is performed only when the MMU is enabled. They can also be disabled for a particular translation base register, in which case the MMU returns a translation fault.

You can program the MMU to make the accesses that are generated by translation table walks cacheable. This means that translation table entries can be cached in the L2 cache, the L3 cache, and external caches.

During a lookup or translation table walk, the access permission bits in the matching translation table entry determine whether the access is permitted. If the permission checks are violated, the MMU signals a permission fault. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

The following figure shows the translation table walk process.

**Figure 6-1: Translation table walks**

In translation table walks the descriptor is fetched from the L2 memory system.

### Related information

[7 L1 instruction memory system](#) on page 58

[8 L1 data memory system](#) on page 61

[9 L2 memory system](#) on page 66

## 6.5 Hardware management of the Access flag and dirty state

The core includes the option to perform hardware updates to the translation tables.

This feature is enabled in TCR\_ELx (where x is 1-3) and VTCR\_EL2. To support hardware management of dirty state, translation table descriptors include the *Dirty Bit Modifier* (DBM) field.

The Cortex®-X2 core supports hardware updates to the Access flag and to dirty state only when the translation tables are held in Inner Write-Back and Outer Write-Back Normal memory regions. If software requests a hardware update in a region that is not Inner Write-Back or Outer Write-Back Normal memory, then the Cortex®-X2 core returns an abort with the following encoding:

- ESR\_ELx.DFSC = 0b110001 for Data Aborts
- ESR\_ELx.IFSC = 0b110001 for Instruction Aborts

## 6.6 Responses

Certain faults and aborts can cause an exception to be taken because of a memory access.

### MMU responses

When one of the following operations is completed, the *Memory Management Unit* (MMU) generates a translation response to the requester:

- An L1 instruction or data *Translation Lookaside Buffer* (TLB) hit

- An L2 TLB hit
- A translation table walk

The responses from the MMU contain the following information:

- The *Physical Address* (PA) that corresponds to the translation
- A set of permissions
- Secure or Non-secure state information
- All the information that is required to report aborts

## MMU aborts

The MMU can detect faults that are related to address translation and can cause exceptions to be taken to the core. Faults can include address size faults, translation faults, access flag faults, and permission faults.

## External aborts

External aborts occur in the memory system, and are different from aborts that the MMU detects. Normally, external memory aborts are rare. External aborts are caused by errors that are flagged by the external memory interfaces or are generated because of an uncorrected *Error Correcting Code* (ECC) error in the L1 data cache or L2 cache arrays.

External aborts are reported synchronously when they occur during translation table walks, data accesses due to all loads to Normal memory, all loads with acquire semantics and all `AtomicLd`, `AtomicCas`, and `AtomicSwap` instructions. The address captured in the *Fault Address Register* (FAR) is the target address of the instruction that generated the synchronous abort. External aborts are reported asynchronously, then they occur for loads to Device memory without release semantics, stores to any memory type, and `AtomicSt`, cache maintenance, `Tlbi`, and `Ic` instructions.

Cortex®-X2 takes a synchronous abort on a Normal memory `ldrx` that receives a non-EXOK response from CHI. The abort is asynchronous for Device memory `ldrx`. For `strx`, OK and EXOK responses are expected and do not cause aborts. NDErr and DErr responses for `WriteNoSnp Excl=1` cause asynchronous aborts.

## Misprogramming contiguous hints

A programmer might mis-program the translation tables so that:

- The block size being used to translate the address is larger than the size of the input address.
- The address range translated by a set of blocks that is marked as contiguous, by use of the contiguous bit, is larger than the size of the input address.

If there is this kind of mis-programming, the Cortex®-X2 core does not generate a translation fault.

## Conflict aborts

The Cortex®-X2 core does not generate Conflict aborts.

When a TLB conflict is detected in the L1 TLB or L2 TLB, hardware automatically handles the conflict by invalidating the conflict entries.

## 6.7 Memory behavior and supported memory types

The Cortex®-X2 core supports memory types defined in the Armv8-A architecture.

Device memory types have the following attributes:

### G – Gathering

The capability to gather and merge requests together into a single transaction

### R – Reordering

The capability to reorder transactions

### E – Early Write Acknowledgment

The capability to accept early acknowledgment of write transactions from the interconnect



In the following table, the n prefix means the capability is not allowed.

The following table shows how memory types are supported in the Cortex®-X2 core.

**Table 6-2: Supported memory types**

Memory attribute type	Shareability	Inner Cacheability	Outer Cacheability	Notes
Device nGnRnE	Outer Shareable	-	-	Treated as Device nGnRnE
Device nGnRE	Outer Shareable <sup>1</sup>	-	-	Treated as Device nGnRE
Device nGRE	Outer Shareable <sup>1</sup>	-	-	Treated as Device nGRE
Device GRE	Outer Shareable <sup>1</sup>	-	-	Treated as Device GRE
Normal	Outer Shareable <sup>1</sup>	Non-cacheable	Any	Treated as Non-cacheable
Normal	Outer Shareable <sup>1</sup>	Write-Through Cacheable	Any	Treated as Non-cacheable
Normal	Outer Shareable <sup>1</sup>	Write-Back Cacheable	Non-cacheable	Treated as Non-cacheable
Normal	Outer Shareable <sup>1</sup>	Write-Back Cacheable	Write-Through Cacheable	Treated as Non-cacheable
Normal	See <a href="#">Table 6-3: Shareability for Normal memory</a> on page 56.	Write-Back Cacheable (any allocation hint)	Write-Back Cacheable No Allocate	Treated as Write-Back Read and Write Allocate but the outer cacheability propagated to the <i>DynamiQ™ Shared Unit-110</i> (DSU-110) is 0 (No Allocate)

<sup>1</sup> Non-cacheable and Device are treated as Outer Shareable. Combinations of Non-cacheable and Write-Through are treated as Non-cacheable, and therefore are Outer Shareable.

Memory attribute type	Shareability	Inner Cacheability	Outer Cacheability	Notes
Normal	See <a href="#">Table 6-3: Shareability for Normal memory</a> on page 56.	Write-Back Cacheable (any allocation hint)	Write-Back Read or Write Allocate	Treated as Write-Back Read and Write Allocate but the outer cacheability propagated to the DSU-110 is 1, therefore upgraded to Write and Read Allocate

The following table shows how the shareability is treated for certain Normal memory.

**Table 6-3: Shareability for Normal memory**

Shareability	Treated as
Non-shareable	Non-shareable
Outer Shareable	Outer Shareable
Inner Shareable	Outer Shareable

## 6.8 Page-based hardware attributes

*Page-Based Hardware Attributes* (PBHA) is an optional, **IMPLEMENTATION DEFINED** feature.

It allows software to set up to four bits in the translation tables, which are then propagated through the memory system with transactions and can be used in the system to control system components. The meaning of the bits is specific to the system design.

For information on how to set and enable the PBHA bits in the translation tables, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*. When disabled, the PBHA value that is propagated on the bus is 0.

For memory accesses caused by a translation table walk, the AHTCR, ATTBCR, and AVTCR registers control the PBHA values.

### PBHA combination between stage 1 and stage 2 on memory accesses

PBHA should always be considered as an attribute of the physical address.

When stage 1 and stage 2 are enabled:

- If both stage 1 PBHA and stage 2 PBHA are enabled, the final PBHA is stage 2 PBHA.
- If stage 1 PBHA is enabled and stage 2 PBHA is disabled, the final PBHA is stage 1 PBHA.
- If stage 1 PBHA is disabled and stage 2 PBHA is enabled, the final PBHA is stage 2 PBHA.
- If both stage 1 PBHA and stage 2 PBHA are disabled, the final PBHA is defined to 0.

Enable of PBHA has a granularity of 1 bit, so this property is applied independently on each PBHA bit.



## Mismatched aliases

If the same physical address is accessed through more than one virtual address mapping, and the PBHA bits are different in the mappings, then the results are **UNPREDICTABLE**. The PBHA value sent on the bus could be for either mapping.

## 7 L1 instruction memory system

The Cortex®-X2 L1 memory system is responsible for fetching instructions and predicting branches. It includes the L1 instruction cache, the L1 instruction *Translation Lookaside Buffer* (TLB), and the *Macro-operation* (MOP) cache.

The L1 instruction memory system provides an instruction stream to the decoder. To increase overall performance and reduce power consumption, the L1 instruction memory system uses dynamic branch prediction and instruction caching.

The following table shows the L1 instruction memory system features.

**Table 7-1: L1 instruction memory system features**

Feature	Description
L1 instruction cache	64KB
	4-way set associative
	<i>Virtually Indexed, Physically Tagged</i> (VIPT) behaving as <i>Physically Indexed, Physically Tagged</i> (PIPT)
	Always protected with parity
Cache line length	64 bytes
<i>Macro-operation</i> (MOP) cache	3072 Macro-operations
	4-way skewed associative
	<i>Virtually Indexed, Virtually Tagged</i> (VIVT) behaving as <i>Physically Indexed, Physically Tagged</i> (PIPT)
	Level 0 instruction cache working in the fetch stages of the pipeline to improve throughput and latency
Cache policy	<i>Pseudo-Least Recently Used</i> (LRU) cache replacement policy



The L1 instruction TLB also resides in the L1 instruction memory system. However, it is part of the *Memory Management Unit* (MMU) and is described in [6 Memory management](#) on page 49.

### 7.1 L1 instruction cache behavior

The L1 instruction cache is invalidated automatically at reset unless the core power mode is initialized to Debug Recovery.

In Debug Recovery mode, the L1 instruction cache is not functional.

#### L1 instruction cache disabled behavior

If the L1 instruction cache is disabled, then instruction fetches cannot access any of the instruction cache arrays, except for cache maintenance operations which can execute normally.

If the L1 instruction cache is disabled, then all instruction fetches to cacheable memory are treated as if they were non-cacheable. This treatment means that instruction fetches might not be coherent with caches in other cores, and software must take this into account.



No relationship between cache sets and *Physical Address* (PA) can be assumed. Arm recommends that cache maintenance operations by set/way are used only to invalidate the entire cache.

## Related information

[5.4.5 Debug recovery mode](#) on page 44

## 7.2 L1 instruction cache Speculative memory accesses

Instruction fetches are Speculative and there can be several unresolved branches in the pipeline.

A branch instruction or exception in the code stream can cause a pipeline flush, discarding the currently fetched instructions. On instruction fetches, pages with Device memory type attributes are treated as Non-Cacheable Normal Memory.

Device memory pages must be marked with the translation table descriptor attribute bit *eXecute Never* (XN). The device and code address spaces must be separated in the physical memory map. This separation prevents Speculative fetches to read-sensitive devices when address translation is disabled.

If the L1 instruction cache is enabled and if the instruction fetches miss in the L1 instruction cache, then they can still look up in the L1 data cache. However, the lookup never causes an L1 data cache refill, regardless of the data cache enable status. The line is only allocated in the L2 cache, provided that the L1 instruction cache is enabled.

## 7.3 Program flow prediction

The Cortex®-X2 core contains program flow prediction hardware, also known as branch prediction. Branch prediction increases overall performance and reduces power consumption.

Program flow prediction is enabled when the *Memory Management Unit* (MMU) is enabled for the current exception level. If program flow prediction is disabled, then all taken branches incur a penalty that is associated with cleaning the pipeline. If program flow prediction is enabled, then it predicts whether a conditional or unconditional branch is to be taken, as follows:

- For conditional branches, it predicts whether the branch is to be taken and the address to which the branch goes, known as the branch target address.
- For unconditional branches, it only predicts the branch target address.

Program flow prediction hardware contains the following functionality:

- A *Branch Target Buffer* (BTB) holding the branch target address of previously observed taken branches
- A branch direction predictor that uses the previous branch history
- The return stack, a stack of nested subroutine return addresses
- A static branch predictor
- An indirect branch predictor

### Predicted and non-predicted instructions

Unless otherwise specified, the following list applies to A64 instructions. Program flow prediction hardware predicts all branch instructions, and includes:

- Conditional branches
- Unconditional branches
- Indirect branches that are associated with procedure call and return instructions

Exception return branch instructions are not predicted.

### Return stack

The return stack stores the address and instruction set state. This address is equal to the link register value stored in X30 in AArch64 state.

In AArch64, any of the following instructions causes a return stack push:

- BL
- BLR
- BLRAA
- BLRAAX
- BLRAB
- BLRABZ

Any of the following instructions cause a return stack pop:

- RET
- RETAA
- RETAB

The following instructions are not predicted:

- ERET
- ERETAA
- ERETAB

## 8 L1 data memory system

The Cortex®-X2 L1 data memory system is responsible for executing load and store instructions, as well as specific instructions such as atomics, cache maintenance operations, and memory tagging instructions. It includes the L1 data cache and the L1 data *Translation Lookaside Buffer* (TLB).

The L1 data memory system executes load and store instructions and services memory coherency requests.

The following table shows the L1 data memory system features.

**Table 8-1: L1 data memory system features**

Feature	Description
L1 data cache	64KB
	4-way set associative
	<i>Virtually indexed, Physically Tagged</i> (VIPT) behaving as <i>Physically Indexed, Physically Tagged</i> (PIPT)
	Always protected with <i>Error Correcting Code</i> (ECC)
Cache line length	64 bytes
Cache policy	Pseudo- <i>Least Recently Used</i> (LRU) cache replacement policy
Interface with integer execute pipeline and vector execute	<ul style="list-style-type: none"> <li>3×64-bit read paths and 4×64-bit write paths for the integer execute pipeline</li> <li>3×128-bit read paths and 2×128-bit write paths for the vector execute pipeline</li> </ul>



The L1 data TLB also resides in the L1 data memory system. However, it is part of the *Memory Management Unit* (MMU) and is described in [6 Memory management](#) on page 49.

### 8.1 L1 data cache behavior

The L1 data cache is invalidated automatically at reset unless the core power mode is initialized to Debug recovery.

In Debug recovery mode, the L1 data cache is not functional.

There is no operation to invalidate the entire data cache. If software requires this function, then it must be constructed by iterating over the cache geometry and executing a series of individual invalidates by set/way instructions. DCCISW operations perform both a clean and invalidate of the target set/way. The values of HCR\_EL2.SWIO have no effect.

#### L1 data cache disabled behavior

If the L1 data cache is disabled, then:

- A new line is not allocated in the L2 or L3 caches as a result of an instruction fetch.

- All load and store instructions to cacheable memory are treated as Non-cacheable.
- Data cache maintenance operations continue to execute normally.

The L1 data and L2 caches cannot be disabled independently. When a core disables the L1 data cache, cacheable memory accesses issued by that core are no longer cached in the L1 or L2 cache.

To maintain data coherency between multiple cores, the Cortex®-X2 core uses the *Modified Exclusive Shared Invalid* (MESI) protocol.

### Related information

[5.4.5 Debug recovery mode](#) on page 44

## 8.2 Instruction implementation in the L1 data memory system

The Cortex®-X2 core supports the atomic instructions added in the Arm®v8.1-A architecture. Atomic instructions to Cacheable memory can be performed as either near atomics or far atomics, depending on where the cache line containing the data resides.

If an instruction hits in the L1 data cache, then the Cortex®-X2 core tries to perform it as a near atomic. Then, based on system behavior, the core can decide to perform it as a far atomic.

If the operation misses everywhere within the cluster and the interconnect supports far atomics, then the atomic is passed on to the interconnect to perform the operation. If the operation hits anywhere inside the cluster, or if an interconnect does not support atomics, then the L3 memory system performs the atomic operation. If the line is not already there, it allocates the line into the L3 cache.

Therefore if software prefers that the atomic is performed as a near atomic, then precede the atomic instruction with a `PLDW` or `PRFM PSTL1KEEP` instruction. Alternatively, `CPUECTLR` can be programmed such that different types of atomic instructions attempt to execute as a near atomic. One cache fill is made on an atomic. If the cache line is lost before the atomic operation can be made, then it is sent as a far atomic.

The Cortex®-X2 core supports atomics to Device or Non-cacheable memory, however this relies on the interconnect also supporting atomics. If such an atomic instruction is executed when the interconnect does not support them, then it results in an abort.

## 8.3 Internal exclusive monitor

The Cortex®-X2 core includes an internal exclusive monitor with a 2-state, open and exclusive state machine that manages Load-Exclusive and Store-Exclusive accesses and Clear-Exclusive (CLREX) instructions.

You can use these instructions to construct semaphores, ensuring synchronization between different processes running on the core, and also between different cores that are using the same coherent memory locations for the semaphore. A Load-Exclusive instruction tags a small block of memory for exclusive access. CTR\_ELO defines the size of the tagged blocks as 16 words, one cache line.



A load/store exclusive instruction is, in the A64 instruction set, any instruction that has a mnemonic starting with `LDX`, `LDAX`, `STX`, or `STLX`.

See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for more information on these instructions.

## 8.4 Data prefetching

Data prefetching can boost execution performance by fetching data before it is needed.

### Preload instructions

The Cortex®-X2 core supports the AArch64 prefetch memory instructions, `PRFM`.

These instructions signal to the memory system that memory accesses from a specified address are likely to occur soon. The memory system takes actions that aim to reduce the latency of memory accesses when they occur.

`PRFM` instructions perform a lookup in the cache. If they miss and are to a cacheable address, then a linefill starts. However, a `PRFM` instruction retires when its linefill is started, and it does not wait until the linefill is complete.

The *Preload Instruction* (`PLI`) memory system hint performs preloading in the L2 cache for cacheable accesses if they miss in the L2 cache. Instruction preloading is performed in the background.

For more information about prefetch memory and preloading caches, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

### Data prefetching and monitoring

The load/store unit includes a hardware prefetcher that is responsible for generating prefetches targeting both the L1 and the L2 caches. The load side prefetcher uses the *Virtual Address* (VA) to

prefetch to both the L1 and L2 caches. The store side prefetcher uses the *Physical Address* (PA), and only prefetches to the L2 cache.

The [A.1.15 IMP\\_CPUCTLR\\_EL1, CPU Extended Control Register](#) on page 159 allows control over the prefetcher.

### Data cache zero

In the Cortex®-X2 core, the *Data Cache Zero by Virtual Address* (DC ZVA) instruction enables a block of 64 bytes in memory, aligned to 64 bytes in size, to be set to zero.

For more information, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

## 8.5 Write streaming mode

The Cortex®-X2 core supports write streaming mode, sometimes referred to as read allocate mode, both for the L1 and the L2 cache.

A cache line is allocated to the L1 or L2 cache on either a read miss or a write miss. However, writing large blocks of data can pollute the cache with unnecessary data. It can also waste power and performance when a linefill is performed only to discard the linefill data because the entire line was subsequently written by the `memset()`. In some situations, cache line allocation on writes is not required. For example, when executing the C standard library `memset()` function to clear a large block of memory to a known value.

To prevent unnecessary cache line allocation, the *Bus Interface Unit* (BIU) can detect when the core has written a full cache line before the linefill completes. If this situation is detected on a configurable number of consecutive linefills, then it switches into write streaming mode.

When in write streaming mode, load operations behave as normal, and can still cause linefills. Writes still lookup in the cache, but if they miss then they write out to the L2 or L3 cache rather than starting a linefill.



More than the specified number of linefills might be observed on the master interface, before the BIU switches to write streaming mode.

---

The BIU continues in write streaming mode until either:

- It detects a cacheable write burst that is not a full cache line.
- There is a load operation from the same line that is being written to the L2 or the L3 cache.

When a Cortex®-X2 core has switched to write streaming mode, the BIU continues to monitor the bus traffic. It signals to the L2 or L3 cache to go into write streaming mode when it observes a further number of full cache line writes.



The write streaming threshold defines the number of consecutive cache lines that are fully written without being read before store operations stop causing cache allocations. You can configure the write streaming threshold for each cache:

- IMP\_CPUECTLR\_EL1.L1WSCTL configures the L1 write streaming mode threshold.
- IMP\_CPUECTLR\_EL1.L2WSCTL configures the L2 write streaming mode threshold.
- IMP\_CPUECTLR\_EL1.L3WSCTL configures the L3 write streaming mode threshold.

**Related information**

[A.1.15 IMP\\_CPUECTLR\\_EL1, CPU Extended Control Register](#) on page 159

## 9 L2 memory system

The Cortex®-X2 L2 memory system connects the core with the *DynamIQ™ Shared Unit-110* (DSU-110) through the CPU bridge. It includes the L2 *Translation Lookaside Buffer* (TLB) and private L2 cache.

The L2 cache is unified and private to each Cortex®-X2 core in a cluster.

The following table shows the L2 memory system features.

**Table 9-1: L2 memory system features**

Feature	Type
L2 cache	512KB or 1MB
	8-way set associative, 4 banks
	<i>Physically Indexed, Physically Tagged</i> (PIPT)
	Always protected with <i>Error Correcting Code</i> (ECC)
Cache line length	64 bytes
Cache policy	Dynamic biased cache replacement policy
Interface with the DSU-110	One CHI Issue E compliant interface with 256-bit read and write DAT channel widths

### 9.1 L2 cache

The integrated L2 cache handles both instruction and data requests from the instruction and data side, as well as translation table walk requests.

The L1 instruction cache and L2 cache are weakly inclusive. Instruction fetches that miss in the L1 instruction cache and L2 cache allocate both caches, but the invalidation of the L2 cache does not cause back-invalidates of the L1 instruction cache. The L1 data cache and L2 cache are strictly inclusive. Any data contained in the L1 data cache is also present in the L2 cache. Victimization of L2 data can cause invalidations of the L1 data cache.

The L2 cache is invalidated automatically at reset unless the core power mode is initialized to Debug Recovery.

#### Related information

[5.4.5 Debug recovery mode](#) on page 44

### 9.2 Support for memory types

The Cortex®-X2 core simplifies coherency logic by downgrading some memory types.

Memory that is marked as both Inner Write-Back Cacheable and Outer Write-Back Cacheable is cached in the L1 data cache and the L2 cache.

Memory that is marked as Inner Write-Through is downgraded to Non-cacheable.

Memory that is marked Outer Write-Through or Outer Non-cacheable is downgraded to Non-cacheable, even if the inner attributes are Write-Back Cacheable.

The additional attribute hints are used as follows:

#### Allocation hint

Allocation hints help to determine the rules of allocation of newly fetched lines in the system.

#### Transient hint

All cacheable reads and writes allocate into the L1 cache and thus the L2 cache due to inclusivity.

An allocating read to the L1 data cache that has the transient bit set is allocated in the L1 cache. Such reads are marked as most likely to be evicted, according to the L1 eviction policy. Transient lines evicted from the L2 cache do not allocate downstream caches.

## 9.3 Transaction capabilities

The CHI Issue E interface between the Cortex®-X2 L2 memory system and the *DynamiQ™ Shared Unit-110* (DSU-110) provides transaction capabilities for the core.

The following table shows the maximum possible values for read, write, *Distributed Virtual Memory* (DVM) issuing, and snoop capabilities of the Cortex®-X2 L2 cache.

**Table 9-2: Cortex®-X2 transaction capabilities**

Attribute	Maximum value	Description
Write issuing capability	68, 76, 84, or 92	This is the maximum number of outstanding write transactions.  It depends on the configured Transaction Queue (TQ) size: 72, 80, 88, or 96.
Read issuing capability	68, 76, 84, or 92	This is the maximum number of outstanding read transactions.  It depends on the configured TQ size: 72, 80, 88, or 96.
Snoop acceptance capability	41, 45, 49, or 53	This is the maximum number of outstanding snoops accepted.  It depends on the configured TQ size: 72, 80, 88, or 96.
DVM issuing capability	68, 76, 84, or 92	This is the maximum number of outstanding DVM operation transactions.  It depends on the configured TQ size: 72, 80, 88, or 96.

## 10 Direct access to internal memory

The Cortex®-X2 core provides a mechanism to read the internal memory that the L1 and L2 caches and TLB structures use, through **IMPLEMENTATION DEFINED** system registers. This functionality can be useful when investigating issues where the coherency between the data in the cache and data in system memory is broken.



It is not possible to update the contents of the caches or TLB structures.

Direct access to internal memory is available only in EL3. In all other modes, executing these instructions results in an Undefined Instruction exception. There are read-only (RO) registers used to access the contents of the internal memory. The internal memory is selected by programming the **IMPLEMENTATION DEFINED** RAMINDEX register. The following table shows the registers that are used to read the data.

**Table 10-1: System registers used to access internal memory**

Register name	Function	Access	Operation	Rd Data
IMP_DDATA0_EL3	Data Register 0	RO	MRS <Xd>, S3_6_c15_c1_0	Data
IMP_DDATA1_EL3	Data Register 1	RO	MRS <Xd>, S3_6_c15_c1_1	Data
IMP_DDATA2_EL3	Data Register 1	RO	MRS <Xd>, S3_6_c15_c1_2	Data
IMP_IDATA0_EL3	Instruction Register 0	RO	MRS <Xd>, S3_6_c15_c0_0	Data
IMP_IDATA1_EL3	Instruction Register 1	RO	MRS <Xd>, S3_6_c15_c0_1	Data
IMP_IDATA2_EL3	Instruction Register 2	RO	MRS <Xd>, S3_6_c15_c0_2	Data

### 10.1 L1 cache encodings

Both the L1 data and instruction caches are 4-way set associative.

The size of the configured cache determines the number of sets in each way. The encoding that is used to locate the cache data entry for tag and data memory is set in  $x_n$  in the appropriate sys instruction. It is similar for both the tag and data RAM access.

The following tables show the encodings required for locating and selecting a given cache line.

**Table 10-2: Cortex®-X2 L1 instruction cache tag location encoding**

Bit field of $X_n$	Description
[31:24]	RAMID = 0x00
[23:20]	Reserved
[19:18]	Way
[17:14]	Reserved

Bit field of Xn	Description
[13:6]	Virtual address [13:6]
[5:0]	Reserved

**Table 10-3: Cortex®-X2 L1 instruction cache data location encoding**

Bit field of Xn	Description
[31:24]	RAMID = 0x01
[23:20]	Reserved
[19:18]	Way
[17:14]	Reserved
[13:3]	Virtual address [13:3]
[2:0]	Reserved

**Table 10-4: Cortex®-X2 L1 BTB data location encoding**

Bit field of Xn	Description
[31:24]	RAMID = 0x02
[23:16]	Reserved
[15:4]	Index [11:0]
[3:0]	Reserved

**Table 10-5: Cortex®-X2 L1 GHB data location encoding**

Bit field of Xn	Description
[31:24]	RAMID = 0x03
[23:16]	Reserved
[15:5]	Index [10:0]
[4:0]	Reserved

**Table 10-6: Cortex®-X2 L1 instruction TLB data location encoding**

Bit field of Xn	Description
[31:24]	RAMID = 0x04
[23:8]	Reserved
[7:0]	TLB entry (0-47)

**Table 10-7: Cortex®-X2 BIM data location encoding**

Bit field of Xn	Description
[31:24]	RAMID = 0x05
[23:15]	Reserved
[14:4]	Index [10:0]
[3:0]	Reserved

**Table 10-8: Cortex®-X2 L0 Macro-operation cache data location encoding**

Bit field of Xn	Description
[31:24]	RAMID = 0x06
[23:11]	Reserved
[10:0]	Index [10:0]

**Table 10-9: Cortex®-X2 L1 data cache tag location encoding**

Bit field of Xn	Description
[31:24]	RAMID = 0x08
[23:20]	Reserved
[19:18]	Way
[17:16]	Copy: <b>0b00</b> Tag RAM associated with Pipe 0 <b>0b01</b> Tag RAM associated with Pipe 1 <b>0b10</b> Tag RAM associated with Pipe 2 <b>0b11</b> Reserved
[15:14]	Reserved
[13:6]	Physical address [13:6]
[5:0]	Reserved

**Table 10-10: Cortex®-X2 L1 data cache data location encoding**

Bit field of Xn	Description
[31:24]	RAMID = 0x09
[23:20]	Reserved
[19:18]	Way
[17:16]	BankSel
[15:14]	Unused
[13:6]	Physical address [13:6]
[5:0]	Reserved

**Table 10-11: Cortex®-X2 L1 data TLB location encoding**

Bit field of Xn	Description
[31:24]	RAMID = 0x0A
[23:6]	Reserved
[5:0]	TLB Entry (0-47)

### 10.1.1 L1 instruction tag RAM returned data

For each register, any access to the L1 instruction tag RAM returns data.

The following tables show the L1 instruction cache tag format for instruction registers.

**Table 10-12: L1 instruction cache tag format for Instruction Register 0**

Bit field	Description
[31]	Non-secure identifier for the physical address
[30:3]	Physical address [39:12]
[2:1]	Instruction state [1:0]  <b>0b00</b> Invalid  <b>0b01</b> RFU  <b>0b10</b> RFU  <b>0b11</b> A64
[0]	Parity

**Table 10-13: L1 instruction cache tag format for Instruction Register 1**

Bit field	Description
[63:0]	0

**Table 10-14: L1 instruction cache tag format for Instruction Register 2**

Bit field	Description
[63:0]	0

### 10.1.2 L1 instruction data RAM returned data

For each register, any access to the L1 instruction data RAM returns data.

The following tables show the L1 instruction cache data format for instruction registers.

**Table 10-15: L1 instruction cache data format for Instruction Register 0**

Bit field	Description
[63:0]	Data [63:0]

**Table 10-16: L1 instruction cache data format for Instruction Register 1**

Bit field	Description
[63:9]	0
[8:0]	Data [72:64]

**Table 10-17: L1 instruction cache data format for Instruction Register 2**

Bit field	Description
[63:0]	0

### 10.1.3 L1 BTB RAM returned data

For each register, any access to the L1 *Branch Target Buffer* (BTB) RAM returns data.

The following tables show the L1 BTB cache format for instruction registers.

**Table 10-18: L1 BTB cache format for Instruction Register 0**

Bit field	Description
[63:0]	Data [63:0]

**Table 10-19: L1 BTB cache format for Instruction Register 1**

Bit field	Description
[63:28]	0
[27:0]	Data [91:64]

**Table 10-20: L1 BTB cache format for Instruction Register 2**

Bit field	Description
[63:0]	0

### 10.1.4 L1 GHB RAM returned data

For each register, any access to the L1 *Global History Buffer* (GHB) RAM returns data.

The following tables show the L1 GHB cache format for instruction registers.

**Table 10-21: L1 GHB cache format for Instruction Register 0**

Bit field	Description
[63:0]	Data [63:0]

**Table 10-22: L1 GHB cache format for Instruction Register 1**

Bit field	Description
[63:0]	Data [127:64]

**Table 10-23: L1 GHB cache format for Instruction Register 2**

Bit field	Description
[63:0]	0



### 10.1.5 L1 BIM RAM returned data

For each register, any access to the L1 *Bimodal Predictor* (BIM) RAM returns data.

The following tables show the L1 BIM cache format for instruction registers.

**Table 10-24: L1 BIM cache format for Instruction Register 0**

Bit field	Description
[63:16]	0
[15:0]	Data [15:0]

**Table 10-25: L1 BIM cache format for Instruction Register 1**

Bit field	Description
[63:0]	0

**Table 10-26: L1 BIM cache format for Instruction Register 2**

Bit field	Description
[63:0]	0

### 10.1.6 L1 instruction TLB returned data

For each register, any access to the L1 instruction TLB returns data.

The following tables show the L1 instruction TLB format for instruction registers.

**Table 10-27: L1 instruction TLB format for Instruction Register 0**

Bit field	Description
[63:61]	Virtual address [17:12]
[60:59]	PBHA [1:0]
[58]	TLB attribute

Bit field	Description
[57:55]	<p>Memory attributes:</p> <p><b>0b000</b> Device nGnRnE</p> <p><b>0b001</b> Device nGnRE</p> <p><b>0b010</b> Device nGRE</p> <p><b>0b011</b> Device GRE</p> <p><b>0b100</b> Non-cacheable</p> <p><b>0b101</b> Write-Back No-Allocate</p> <p><b>0b110</b> Write-Back Transient</p> <p><b>0b111</b> Write-Back Read-Allocate and Write-Allocate</p>
[54:52]	<p>Page size:</p> <p><b>0b000</b> 4KB</p> <p><b>0b001</b> 16KB</p> <p><b>0b010</b> 64KB</p> <p><b>0b100</b> 2MB</p> <p><b>Other</b> Reserved</p>
[51:48]	TLB attribute
[47]	Outer-shared
[46]	Inner-shared
[45:40]	TLB attribute
[39:24]	ASID[15:0]
[23:8]	VMID[15:0]

Bit field	Description
[7:5]	MSID[2:0]:  <b>0b000</b> Secure EL1/EL0  <b>0b001</b> Secure EL2  <b>0b101</b> Secure EL3  <b>0b010</b> Non-secure EL1/EL0  <b>0b011</b> Non-secure EL2
[4:1]	TLB attribute
[0]	Valid

**Table 10-28: L1 instruction TLB format for Instruction Register 1**

Bit field	Description
[63]	TLB attribute
[62]	Non-secure
[61:34]	Physical address [39:12]
[33:0]	Virtual address [48:15]

**Table 10-29: L1 instruction TLB format for Instruction Register 2**

Bit field	Description
[63:1]	Reserved
[0]	TLB attribute

### 10.1.7 L0 macro-operation RAM returned data

For each register, any access to the L0 *Macro-operation* (MOP) RAM returns data.

The following tables show the L0 MOP cache format for instruction registers.

**Table 10-30: L0 MOP cache format for Instruction Register 0**

Bit field	Description
[63:0]	Macro-operation data [63:0]

**Table 10-31: L0 MOP cache format for Instruction Register 1**

Bit field	Description
[63:28]	0
[27:0]	Macro-operation data [103:64]

**Table 10-32: L0 MOP cache format for Instruction Register 2**

Bit field	Description
[63:0]	0

### 10.1.8 L1 data tag RAM returned data

For each register, any access to the L1 data tag RAM returns data.

The following tables show the L1 data cache tag format for data registers.

**Table 10-33: L1 data cache tag format for Data Register 0**

Bit field	Description
[63]	0
[62:56]	ECC
[55:27]	Non-secure identifier, physical address [39:12]
[26]	Origin
[25]	Prefetch
[24]	Transient/WBNA
[23:20]	<i>Memory Tagging Extension (MTE)</i> tag poison
[19:4]	MTE tag data
[3:2]	MTE tag state:  <b>0b00</b> Invalid  <b>0b01</b> Shared  <b>0b11</b> Dirty
[1:0]	<i>Modified Exclusive Shared Invalid (MESI)</i> :  <b>0b00</b> Invalid  <b>0b01</b> Shared  <b>0b10</b> Exclusive  <b>0b11</b> Modified

**Table 10-34: L1 data cache tag format for Data Register 1**

Bit field	Description
[63:0]	0

**Table 10-35: L1 data cache tag format for Data Register 2**

Bit field	Description
[63:0]	0

### 10.1.9 L1 data data RAM returned data

For each register, any access to the L1 data data RAM returns data.

The following tables show the L1 data cache data format for data registers.

**Table 10-36: L1 data cache data format for Data Register 0**

Bit field	Description
[63:0]	word1_data[31:0], word0_data[31:0]

**Table 10-37: L1 data cache data format for Data Register 1**

Bit field	Description
[63:0]	word3_data[31:0], word2_data[31:0]

**Table 10-38: L1 data cache data format for Data Register 2**

Bit field	Description
[63:32]	0
[31:0]	word3_ecc [6:0], word3_poison, word2_ecc [6:0], word2_poison, word1_ecc [6:0], word1_poison, word0_ecc [6:0], word0_poison

### 10.1.10 L1 data TLB returned data

For each register, any access to the L1 data TLB returns data.

The following tables show the L1 data TLB format for data registers.

**Table 10-39: L1 data TLB format for Data Register 0**

Bit field	Description
[63]	Virtual address [12]
[62:61]	LOR ID [1:0]
[60]	LOR match
[59]	Outer-shared
[58]	Inner-shared
[57:56]	S1 translation regime [1:0]
[55:54]	S2 translation regime [1:0]

Bit field	Description
[53:51]	<p>Memory attributes [2:0]:</p> <p><b>0b000</b> Device nGnRnE</p> <p><b>0b001</b> Device nGnRE</p> <p><b>0b010</b> Device nGRE</p> <p><b>0b011</b> Device GRE</p> <p><b>0b100</b> Non-cacheable</p> <p><b>0b101</b> Write-Back No-Allocate</p> <p><b>0b110</b> Write-Back Transient</p> <p><b>0b111</b> Write-Back Read-Allocate and Write-Allocate</p>
[50]	Outer allocate
[49]	S2 DBM bit
[48]	S1 DBM bit
[47]	TLB coalesced bit
[46:43]	Permission bit [3:0]
[42]	Device/Non-cacheable HTRAP
[41]	nG bit
[40]	Smash bit
[39:37]	<p>Page size [2:0]:</p> <p><b>0b000</b> 4KB</p> <p><b>0b001</b> 16KB</p> <p><b>0b010</b> 64KB</p> <p><b>0b011</b> Reserved</p> <p><b>0b100</b> 2MB</p> <p><b>0b101</b> Reserved</p> <p><b>0b110</b> 512MB</p> <p><b>0b111</b> Reserved</p>

Bit field	Description
[36]	Non-secure
[35:33]	MSID [2:0]
[32:17]	ASID [15:0]
[16:1]	VMID [15:0]
[0]	Valid

**Table 10-40: L1 data TLB format for Data Register 1**

Bit field	Description
[63:36]	Physical address [39:12]
[35:0]	Virtual address [48:13]

**Table 10-41: L1 data TLB format for Data Register 2**

Bit field	Description
[63:6]	Reserved
[5]	Tagged MTE
[4]	FWB override
[3:0]	PBHA [3:0]

## 10.2 L2 cache encodings

The L2 cache is 8-way set associative.

The size of the configured cache determines the number of sets in each way. The encoding that is used to locate the cache data entry for tag and data memory is set in  $x_n$  in the appropriate `sys` instruction. It is similar for both the tag and data RAM access.

The following tables show the encodings required for locating and selecting a given cache line.

**Table 10-42: Cortex®-X2 L2 cache tag location encoding for 512KB<sup>2</sup>**

Bit field of $X_n$	Description
[31:24]	RAMID = 0x10
[23:21]	Reserved
[20:18]	Way (0-7)
[17:16]	Reserved
[15:13]	Index [15:13]
[12:10]	XOR(Index [12:10], Way[2:0])
[9:8]	Index [9:8]
[7:6]	XOR(Index [7:6], Index [11:10], Way[1:0])

<sup>2</sup> index[15:8]=XOR(physical address[15:8], physical address[23:16]) index[7:6]=XOR(physical address[7:6], physical address[11:10])

Bit field of Xn	Description
[5:0]	Reserved

**Table 10-43: Cortex®-X2 L2 cache tag location encoding for 1MB<sup>3</sup>**

Bit field of Xn	Description
[31:24]	RAMID = 0x10
[23:21]	Reserved
[20:18]	Way (0-7)
[17]	Reserved
[16:12]	Index [16:12]
[11:9]	XOR(Index [11:9], Way[2:0])
[8]	Index [8]
[7:6]	XOR(Index [7:6], Index [11:10], Way[2:1])
[5:0]	Reserved

**Table 10-44: Cortex®-X2 L2 cache data location encoding for 512KB**

Bit field of Xn	Description
[31:24]	RAMID = 0x11
[23:21]	Reserved
[20:18]	Way (0-7)
[17:16]	Reserved
[15:13]	Index [15:13]
[12:10]	XOR(Index [12:10], Way[2:0])
[9:8]	Index [9:8]
[7:6]	XOR(Index [7:6], Index [11:10], Way[1:0])
[5:4]	Physical address [5:4]
[3:0]	Reserved

**Table 10-45: Cortex®-X2 L2 cache data location encoding for 1MB**

Bit field of Xn	Description
[31:24]	RAMID = 0x11
[23:21]	Reserved
[20:18]	Way (0-7)
[17]	Reserved
[16:12]	XOR(Index [16:12], 5'b01000)
[11:9]	XOR(Index [11:9], Way[2:0])
[8]	Index [8]
[7:6]	XOR(Index [7:6], Index [11:10], Way[2:1])
[5:4]	Physical address [5:4]

<sup>3</sup> index[16:8]=XOR(physical address[16:8], physical address[25:17]), index[7:6]=XOR(physical address[7:6], physical address[11:10])



Bit field of Xn	Description
[3:0]	Reserved

**Table 10-46: Cortex®-X2 L2 TLB location encoding**

Bit field of Xn	Description
[31:24]	RAMID = 0x18
[23:21]	RES0
[20:18]	Way (0-4)
[17:8]	RES0
[7:0]	TLB entry (0-255)

**Table 10-47: Cortex®-X2 L2 victim location encoding**

Bit field of Rd	Description
[31:24]	RAMID = 0x12
[23:17]	Reserved
[16]	Index [16]
[15:8]	Index [15:8]
[7:6]	XOR(Index[11:10], Index[7:6])
[5:0]	Reserved

### 10.2.1 L2 tag RAM returned data

For each register, any access to the L2 tag RAM returns data.

The following tables show the L2 tag cache format for data registers. In the first table:

**For 512KB L2 cache**

n=33, m=16

**For 1MB L2 cache**

n=32, m=17

**Table 10-48: L2 tag cache format for Data Register 0**

Bit field	Description
[63:n+22]	0
[n+21:n+13]	ECC
[n+12]	MPAM_PMG
[n+11:n+6]	MPAM_PARTID
[n+5]	MPAM_NS
[n+4:n+1]	PBHA[3:0]
[n:10]	Physical tag [39:m]
[9]	Non-secure
[8:7]	Virtual address [13:12]

Bit field	Description
[6]	Shareable
[5]	L1 data cache valid
[4:3]	MTE state:  <b>0b00</b> Invalid  <b>0b10</b> Clean  <b>0b11</b> Dirty
[2:0]	L2 state:  <b>0b101</b> UniqueDirty  <b>0b001</b> UniqueClean  <b>0bx11</b> SharedClean  <b>0bxx0</b> Invalid

Table 10-49: L2 tag cache format for Data Register 1

Bit field	Description
[63:0]	0

Table 10-50: L2 tag cache format for Data Register 2

Bit field	Description
[63:0]	0

## 10.2.2 L2 data RAM returned data

For each register, any access to the L2 data RAM returns data.

The following tables show the L2 data RAM format for instruction registers.

Table 10-51: L2 data RAM format for Data Register 0

Bit field	Description
[63:0]	Data [63:0]

Table 10-52: L2 data RAM format for Data Register 1

Bit field	Description
[63:0]	Data [127:64]

**Table 10-53: L2 data RAM format for Data Register 2**

Bit field	Description
[63:20]	0
[19:4]	[15:8] is ECC for Data [127:64], [7:0] is ECC for Data [63:0]
[3:0]	MTE tags

## 10.2.3 L2 TLB RAM returned data

For each register, any access to the L2 TLB RAM returns data.

The following tables show the L2 TLB format for instruction registers.

**Table 10-54: L2 TLB format for Instruction Register 0**

Bit field	Description
[63]	Outer Shareable
[62]	Inner Shareable
[61]	Outer allocate
[60:58]	Memory attributes: <b>0b000</b> Device nGnRnE <b>0b001</b> Device nGnRE <b>0b010</b> Device nGRE <b>0b011</b> Device GRE <b>0b100</b> Non-cacheable <b>0b101</b> Write-Back No-Allocate <b>0b110</b> Write-Back Transient <b>0b111</b> Write-Back Read-Allocate and Write-Allocate
[57:54]	Reserved

Bit field	Description
[53:20]	Physical address  When bit[6] is 0: <ul style="list-style-type: none"> <li>[53:26] = PA[39:12]</li> <li>[25:20] = Reserved</li> </ul> When bit[6] is 1: <ul style="list-style-type: none"> <li>[53:28] = PA[39:14]</li> <li>[27:26] = PA[13:12] for page 3 (highest memory address)</li> <li>[25:24] = PA[13:12] for page 2</li> <li>[23:22] = PA[13:12] for page 1</li> <li>[21:20] = PA[13:12] for page 0 (lowest memory address)</li> </ul>
[19:17]	Page size: <b>0b000</b> 4KB <b>0b001</b> 16KB <b>0b010</b> 64KB <b>0b100</b> 2MB <b>0b101</b> 32MB <b>0b110</b> 512MB <b>0b111</b> 1GB
[16:7]	Reserved
[6]	Coalesced entry
[5:2]	Valid bits
[1:0]	Reserved

**Table 10-55: L2 TLB format for Instruction Register 1**

Bit field	Description
[63:51]	ASID[12:0]
[50:47]	PBHA
[46]	Walk cache entry
[45:17]	Virtual address [48:20]
[16:13]	Reserved
[12]	Non-secure
[11:1]	Reserved
[0]	nG, indicates a non-global page

**Table 10-56: L2 TLB format for Instruction Register 2**

Bit field	Description
[63:22]	Reserved
[21:19]	MSID[2:0]:  <b>0b000</b> Secure EL1  <b>0b001</b> Secure EL2  <b>0b010</b> Non-secure EL1  <b>0b011</b> Non-secure EL2  <b>0b101</b> EL3
[18:3]	VMID[15:0]
[2:0]	ASID[15:13]

## 10.2.4 L2 Victim RAM returned data

For each register, any access to the L2 victim RAM returns data.

The following tables show the L2 victim RAM format for instruction registers.

**Table 10-57: Cortex®-X2 L2 victim format for data register 0**

Bit field of Rd	Description
[63:56]	Prefetch bit
[55:48]	Data source
[47:40]	Transient bit
[39:32]	Outer allocation hint
[31:24]	Pointer fill counter
[23:0]	Replacement [23:0]

**Table 10-58: Cortex®-X2 L2 victim format for data register 1**

Bit field of Rd	Description
[63:0]	0

**Table 10-59: Cortex®-X2 L2 victim format for data register 2**

Bit field of Rd	Description
[63:0]	0

# 11 RAS Extension support

The Cortex®-X2 core supports the *Reliability, Availability, and Serviceability* (RAS) Extension, including all extensions up to Arm®v9.0-A.

In particular, the Cortex®-X2 core supports:

- Cache protection with *Single Error Correct Double Error Detect* (SECCDED) ECC on the RAMs that contain dirty data. This includes the L1 data tag and data RAMs, the L2 tag and data RAMs, and the L2 *Transaction Queue* (TQ) RAMs.
- Cache protection with *Single Error Detect* (SED) parity on the RAMs that only contain clean data. This includes the L1 instruction tag and data cache, the *Macro-operation* (MOP) cache, and the *Memory Management Unit* (MMU) RAMs.
- The *Error Synchronization Barrier* (ESB) instruction. When an ESB instruction is executed, the core ensures that all SError Interrupts that are generated by instructions before the ESB are either taken by the core or pended in DISR\_EL1.
- Poison attribute on bus transfers
- Error Data Record registers
- *Fault Handling Interrupts* (FHIs)
- *Error Recovery Interrupts* (ERIs)
- Error injection

The Cortex®-X2 core features the following nodes:

- Node 0 that includes the shared L3 memory system in the *DynamiQ™ Shared Unit-110* (DSU-110)
- Node 1 that includes the private L1 and L2 memory systems in the core

For more information on the architectural RAS Extension and the definition of a node, see the *Arm® Reliability, Availability, and Serviceability (RAS) Specification Armv8, for the Armv8-A architecture profile*.

For information on the node that includes the shared L3 memory system, see *RAS Extension Support* in the *Arm® DynamiQ™ Shared Unit-110 Technical Reference Manual*.

## 11.1 Cache protection behavior

The configuration of the *Reliability, Availability, and Serviceability* (RAS) Extension that is implemented in the Cortex®-X2 core includes cache protection. In this case, the Cortex®-X2 core protects against errors that result in a RAM bitcell holding the incorrect value.

The RAMs in the Cortex®-X2 core have the following capability:

**SED parity**

Single Error Detect. One bit of parity is applicable to the entire word. The word size is specific for each RAM and depends on the protection granule.

**SECDED ECC**

Single Error Correct, Double Error Detect. When the datum and code bits are all-zero or all-one, the interpretation is that an error has occurred that the ECC scheme cannot correct. However, it might be corrected by other means, such as refetching cached data.

The following table shows which protection type is applied to each RAM in the Cortex®-X2 core. The core can progress and remain functionally correct when there is a single bit error in any RAM.

**Table 11-1: RAM cache protection**

RAM	ECC or parity
L1 instruction cache data	SED parity
L1 instruction cache tag	SED parity
L0 <i>Macro-operation</i> (MOP) cache data	SED parity
L1 data cache data	SECDED ECC
L1 data cache tag	SECDED ECC
MMU <i>Translation Cache</i> (MMUTC)	SED parity
L2 cache data	SECDED ECC
L2 cache tag	SECDED ECC
L2 <i>Transaction Queue</i> (TQ)	SECDED ECC

If there are multiple single bit errors in different RAMs or within different protection granules within the same RAM, then the core also remains functionally correct.

If there is a double bit error in a single RAM within the same protection granule, then the behavior depends on the RAM:

- For RAMs with SECDED capability, the core detects, reports and may defer the error. If the error is in a cache line containing dirty data, then that data might be lost.
- For RAMs with only SED, the core does not detect a double bit error. This might cause data corruption.

If there are errors that are three or more bits within the same protection granule, then depending on the RAM and the position of the errors within the RAM, the core might or might not detect the errors.

The cache protection feature of the core has a minimal performance impact when no errors are present.

## 11.2 Error containment

The Cortex®-X2 core supports error containment for data errors, which means that detected errors are not silently propagated.

Data errors are propagated using data poisoning to ensure that a consumer is aware of the error. Uncorrectable L1 data cache and L2 cache tag errors are not containable.

Error containment also implies support for poisoning if there is a double error on an eviction. This ensures that the error of the associated data is reported when it is consumed.

Support for the *Error Synchronization Barrier* (ESB) instruction in the core also allows further isolation of imprecise exceptions that are reported when poisoned data is consumed.

## 11.3 Fault detection and reporting

When the Cortex®-X2 core detects a fault, it raises a *Fault Handling Interrupt* (FHI) exception or an *Error Recovery Interrupt* (ERI) exception through the fault or the error signals. FHIs and ERIs are reflected in the *Reliability, Availability, and Serviceability* (RAS) registers, which are updated in the node that detects the errors.

### Fault handling interrupts

When ERR1CTLR.FI is set, all detected Deferred errors, Uncorrected errors, and overflows of the corrected error counters cause an FHI to be generated. When ERR1CTLR.CFI is set, all detected Corrected errors also cause an FHI to be generated.

FHIs from core *n* are signaled using **nCOREFAULTIRQ[n]**.

### Error recovery interrupts

When ERR1CTLR.UI is set, all detected Uncorrected errors that are not deferred generate an ERI.

ERIs from core *n* are signaled using **nCOREERRIRQ[n]**.

### Related information

[11.6 AArch64 RAS register summary](#) on page 90

[A.10.4 ERXCTLR\\_EL1, Selected Error Record Control Register](#) on page 361

[A.10.5 ERXSTATUS\\_EL1, Selected Error Record Primary Status Register](#) on page 364

## 11.4 Error detection and reporting

When the Cortex®-X2 core consumes an error, it raises different exceptions depending on the error type.

The Cortex®-X2 core might raise:

- A *Synchronous External Abort* (SEA)



- An *Asynchronous External Abort* (AEA)
- An *Error Recovery Interrupt* (ERI)

## Error detection and reporting registers

The following registers are provided:

- Error Record Feature Registers, ERR<n>FR. These read-only registers specify various error record settings.
- Error Record Control Registers, ERR<n>CTLR. These registers enable error reporting and also enable various interrupts that are related to errors and faults.
- Error Record Miscellaneous Registers, ERR<n>MISCO-3. These registers record details of the error location and counts.
- Pseudo-fault Generation Feature register, ERR<n>PFGF. This read-only register specifies various error settings.

### 11.4.1 Error reporting and performance monitoring

All detected memory errors, *Error Correcting Code* (ECC) or parity errors, trigger the MEMORY\_ERROR event.

The MEMORY\_ERROR event is counted by the *Performance Monitoring Unit* (PMU) counters if it is selected and the counter is enabled.

In Secure state, the event is counted only if MDCR\_EL3.SPME is asserted. See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for a description of MDCR\_EL3.

## Related information

[17.1 Performance monitors events](#) on page 105

## 11.5 Error injection

Error injection consists of inserting an error in the error detection logic to verify the error handling software.

Error injection uses the error detection and reporting registers to insert errors. The Cortex®-X2 core can inject the following error types:

### Corrected errors

A *Corrected Error* (CE) is generated for a single *Error Correcting Code* (ECC) error on an L1 data cache access.

### Deferred errors

A *Deferred Error* (DE) is generated for a double ECC error on eviction of a cache line from the L1 cache to the L2 cache, or as a result of a snoop on the L1 cache.

## Uncontainable errors

An *Uncontainable Error* (UC) is generated for a double ECC error on the L1 tag RAM following an eviction.

An error can be injected immediately or when a 32-bit counter reaches zero. You can control the value of the counter through the Error Pseudo-fault Generation Countdown Register, ERROPFGCDN. The value of the counter decrements on a per clock cycle basis. See the Arm® *Reliability, Availability, and Serviceability (RAS) Specification Armv8, for the Armv8-A architecture profile* for more information about ERROPFGCDN.



Error injection is a separate source of error within the system and does not create hardware faults.

## 11.6 AArch64 RAS register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** RAS registers in the core. Individual register descriptions provide detailed information.

**Table 11-2: RAS register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
ERRIDR_EL1	3	C5	0	C3	0	See individual bit resets.	64-bit	Error Record ID Register
ERRSELR_EL1	3	C5	0	C3	1	See individual bit resets.	64-bit	Error Record Select Register
ERXFR_EL1	3	C5	0	C4	0	See individual bit resets.	64-bit	Selected Error Record Feature Register
ERXCTLR_EL1	3	C5	0	C4	1	0x0	64-bit	Selected Error Record Control Register
ERXSTATUS_EL1	3	C5	0	C4	2	0x0	64-bit	Selected Error Record Primary Status Register
ERXADDR_EL1	3	C5	0	C4	3	See individual bit resets.	64-bit	Selected Error Record Address Register
ERXPFGF_EL1	3	C5	0	C4	4	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Feature register
ERXPFGCTL_EL1	3	C5	0	C4	5	0x0	64-bit	Selected Pseudo-fault Generation Control register
ERXPFGCDN_EL1	3	C5	0	C4	6	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Countdown register
ERXMISCO_EL1	3	C5	0	C5	0	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 0
ERXMISC1_EL1	3	C5	0	C5	1	0x0	64-bit	Selected Error Record Miscellaneous Register 1
ERXMISC2_EL1	3	C5	0	C5	2	0x0	64-bit	Selected Error Record Miscellaneous Register 2
ERXMISC3_EL1	3	C5	0	C5	3	0x0	64-bit	Selected Error Record Miscellaneous Register 3

## 12 GIC CPU interface

The *Generic Interrupt Controller* (GIC) supports and controls interrupts. The GIC distributor connects to the Cortex®-X2 core through a GIC CPU interface. The GIC CPU interface includes registers to mask, identify, and control the state of interrupts that are forwarded to the core.

Each core in a DynamIQ™-110 cluster has a GIC CPU interface, which connects to a common external distributor component.

The GICv4.1 architecture implemented in the Cortex®-X2 core supports:

- Two Security states
- Secure virtualization
- *Software-Generated Interrupts* (SGIs)
- Message-based interrupts
- System register access for the CPU interface
- Interrupt masking and prioritization
- Cluster environments, including systems that contain more than eight cores
- Wakeup events in power management environments

The GIC includes interrupt grouping functionality that supports:

- Configuring each interrupt to belong to either Group 0 or Group 1, where Group 0 interrupts are always Secure
- Signaling Group 1 interrupts to the target core using either the IRQ or the FIQ exception request. Group 1 interrupts can be Secure or Non-secure
- Signaling Group 0 interrupts to the target core using the FIQ exception request only
- A unified scheme for handling the priority of Group 0 and Group 1 interrupts

See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information about interrupt groups.

### 12.1 Disable the GIC CPU interface

The Cortex®-X2 core always includes the *Generic Interrupt Controller* (GIC) CPU interface. However, you can disable it to meet your requirements.

To disable the GIC CPU interface, assert the **GICCDISABLE** signal HIGH at reset. If you disable it this way, then you can use an external GIC IP to drive the interrupt signals (**nFIQ**, **nIRQ**). If the Cortex®-X2 core is not integrated with an external GIC interrupt distributor component (minimum GICv3 architecture) in the system, then you must disable the GIC CPU interface.

If you disable the GIC CPU interface, then:

- The virtual input signals **nVIRQ** and **nVFIQ** and the input signals **nIRQ** and **nFIQ** can be driven by an external GIC in the SoC.
- GIC system register access generates **UNDEFINED** instruction exceptions.



Note

If you enable the GIC CPU interface, then you must tie off **nVIRQ** and **nVFIQ** to HIGH. This is because the GIC CPU interface generates the virtual interrupt signals to the core. The **nIRQ** and **nFIQ** signals are controlled by software, therefore there is no requirement to tie them HIGH.

See *Functional integration* in the *Arm® DynamIQ™ Shared Unit-110 Configuration and Integration Manual* for more information on these signals.

## 12.2 AArch64 GIC register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** GIC registers in the core. Individual register descriptions provide detailed information.

**Table 12-1: GIC register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
ICC_CTLR_EL1	3	C12	0	C12	4	See individual bit resets.	64-bit	Interrupt Controller Control Register (EL1)
ICV_CTLR_EL1	3	C12	0	C12	4	See individual bit resets.	64-bit	Interrupt Controller Virtual Control Register
ICC_AP0R0_EL1	3	C12	0	C8	4	See individual bit resets.	64-bit	Interrupt Controller Active Priorities Group 0 Registers
ICV_AP0R0_EL1	3	C12	0	C8	4	See individual bit resets.	64-bit	Interrupt Controller Virtual Active Priorities Group 0 Registers
ICC_AP1R0_EL1	3	C12	0	C9	0	See individual bit resets.	64-bit	Interrupt Controller Active Priorities Group 1 Registers
ICV_AP1R0_EL1	3	C12	0	C9	0	See individual bit resets.	64-bit	Interrupt Controller Virtual Active Priorities Group 1 Registers
ICH_VTR_EL2	3	C12	4	C11	1	See individual bit resets.	64-bit	Interrupt Controller VGIC Type Register
ICC_CTLR_EL3	3	C12	6	C12	4	See individual bit resets.	64-bit	Interrupt Controller Control Register (EL3)

## 13 Advanced SIMD and floating-point support

The Cortex®-X2 core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set without floating-point exception trapping. The Cortex®-X2 core floating-point implementation includes Arm®v8.3-A and Arm®v8.5-A features.

The Cortex®-X2 core implements all scalar operations in hardware with support for all combinations of:

- Rounding modes
- Flush-to-zero
- Default *Not a Number* (NaN) modes

## 14 Scalable Vector Extensions support

The Cortex®-X2 core supports the *Scalable Vector Extension* (SVE) and the *Scalable Vector Extension 2* (SVE2). SVE and SVE2 complement and do not replace AArch64 Advanced SIMD and floating-point functionality.

SVE is an optional extension introduced by the Armv8.2 architecture. SVE is supported in AArch64 state only. SVE provides vector instructions that, primarily, support wider vectors than the Arm Advanced SIMD instruction set.

The Cortex®-X2 core implements a scalable vector length of 128 bits.

All the features and additions that SVE introduces are described in the *Arm® Architecture Reference Manual Supplement, The Scalable Vector Extension (SVE), for Armv8-A*.

See the *Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile* for more information about SVE2.

# 15 System control

The system registers control and provide status information for the functions that the core implements.

The main functions of the system registers are:

- System performance monitoring
- Cache configuration and management
- Overall system control and configuration
- *Memory Management Unit* (MMU) configuration and management
- *Generic Interrupt Controller* (GIC) configuration and management

The system registers are accessible in AArch64 execution state at EL0 to EL3.

Some of the system registers are accessible through the external debug interface or Utility bus interface.

## 15.1 AArch64 Identification register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Identification registers in the core. Individual register descriptions provide detailed information.

**Table 15-1: Identification register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
MIDR_EL1	3	C0	0	C0	0	See individual bit resets.	64-bit	Main ID Register
MPIDR_EL1	3	C0	0	C0	5	See individual bit resets.	64-bit	Multiprocessor Affinity Register
REVIDR_EL1	3	C0	0	C0	6	See individual bit resets.	64-bit	Revision ID Register
ID_AA64PFR0_EL1	3	C0	0	C4	0	See individual bit resets.	64-bit	AArch64 Processor Feature Register 0
ID_AA64PFR1_EL1	3	C0	0	C4	1	See individual bit resets.	64-bit	AArch64 Processor Feature Register 1
ID_AA64ZFR0_EL1	3	C0	0	C4	4	See individual bit resets.	64-bit	SVE Feature ID register 0
ID_AA64DFR0_EL1	3	C0	0	C5	0	See individual bit resets.	64-bit	AArch64 Debug Feature Register 0
ID_AA64DFR1_EL1	3	C0	0	C5	1	0x0	64-bit	AArch64 Debug Feature Register 1
ID_AA64AFR0_EL1	3	C0	0	C5	4	0x0	64-bit	AArch64 Auxiliary Feature Register 0
ID_AA64AFR1_EL1	3	C0	0	C5	5	0x0	64-bit	AArch64 Auxiliary Feature Register 1
ID_AA64ISAR0_EL1	3	C0	0	C6	0	See individual bit resets.	64-bit	AArch64 Instruction Set Attribute Register 0
ID_AA64ISAR1_EL1	3	C0	0	C6	1	See individual bit resets.	64-bit	AArch64 Instruction Set Attribute Register 1
ID_AA64MMFR0_EL1	3	C0	0	C7	0	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 0
ID_AA64MMFR1_EL1	3	C0	0	C7	1	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 1
ID_AA64MMFR2_EL1	3	C0	0	C7	2	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 2
CLIDR_EL1	3	C0	1	C0	1	See individual bit resets.	64-bit	Cache Level ID Register

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
GMID_EL1	3	C0	1	C0	4	See individual bit resets.	64-bit	Multiple tag transfer ID register
CTR_ELO	3	C0	3	C0	1	See individual bit resets.	64-bit	Cache Type Register
DCZID_ELO	3	C0	3	C0	7	See individual bit resets.	64-bit	Data Cache Zero ID register
MPAMIDR_EL1	3	C10	0	C4	4	See individual bit resets.	64-bit	MPAM ID Register (EL1)
IMP_CPUCFR_EL1	3	C15	0	C0	0	See individual bit resets.	64-bit	CPU Configuration Register



# 16 Debug

The DynamIQ™-110 cluster provides a debug system that supports both self-hosted and external debug. It has an external DebugBlock component, and integrates various CoreSight debug related components.

The CoreSight debug related components are split into two groups, with some components in the DynamIQ™ cluster, and others in the separate DebugBlock.

The DebugBlock is a dedicated debug component in the DSU-110, separate from the cluster. The DebugBlock operates within a separate power domain, enabling connection to a debugger to be maintained when the cores and the DynamIQ™ cluster are both powered down.

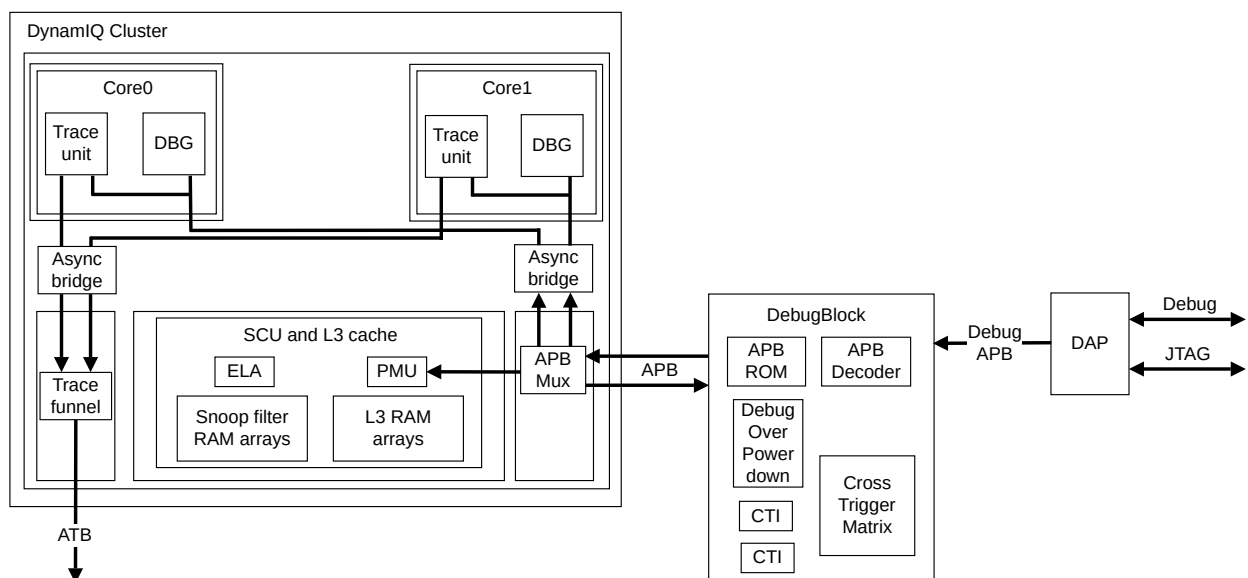
The connection between the cluster and the DebugBlock consists of a pair of *Advanced Peripheral Bus* APB interfaces, one in each direction. All debug traffic, except the authentication interface, takes place over this interface as read or write APB transactions. This debug traffic includes register reads, register writes, and *Cross Trigger Interface* (CTI) triggers.

The debug system implements the following CoreSight debug components:

- Per-core trace unit, integrated into the CoreSight subsystem.
- Per-core CTI, contained in the DebugBlock.
- *Cross Trigger Matrix* (CTM)
- Debug control provided by AMBA® APB interface to the DebugBlock

The following figure shows how the debug system is implemented with the DynamIQ™ cluster.

**Figure 16-1: DynamIQ™ cluster debug components**



The primary debug APB interface on the DebugBlock controls the debug components. The APB decoder decodes the requests on this bus before they are sent to the appropriate component in the DebugBlock or in the DynamIQ™ cluster. The per-core CTIs are connected to a CTM.

Each core contains a debug component that the debug APB bus accesses. The cores support debug over powerdown using modules in the DebugBlock that mirror key core information. These modules allow access to debug over powerdown CoreSight™ registers while the core is powered down.

The trace unit in each core outputs trace, which is funneled in the DynamIQ™ cluster down to a single AMBA® 4 ATBv1.1 interface.

See *Debug* in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual* for more information about the DynamIQ™ cluster debug components.

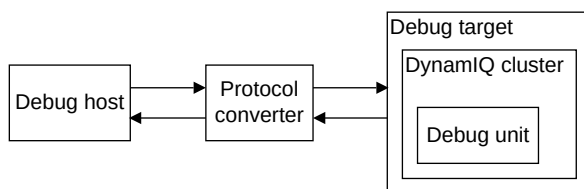
The Cortex®-X2 core also supports direct access to internal memory, that is, cache debug. Direct access to internal memory allows software to read the internal memory that the L1 and L2 cache and *Translation Lookaside Buffer* (TLB) structures use. See [10 Direct access to internal memory](#) on page 68 for more information.

## 16.1 Supported debug methods

The DynamIQ™-110 cluster along with its associated cores is part of a debug system that supports both self-hosted and external debug.

The following figure shows a typical external debug system.

**Figure 16-2: External debug system**



### Debug host

A computer, for example a personal computer, that is running a software debugger such as the Arm® Debugger. You can use the debug host to issue high-level commands. For example, you can set a breakpoint at a certain location or examine the contents of a memory address.

### Protocol converter

The debug host sends messages to the debug target using an interface such as Ethernet. However, the debug target typically implements a different interface protocol. A device such as DSTREAM is required to convert between the two protocols.

## Debug target

The lowest level of the system implements system support for the protocol converter to access the debug unit. For DSU-110 based devices, the mechanism used to access the debug unit is based on the CoreSight architecture. The DSU-110 DebugBlock is accessed using an APB interface and the debug accesses are then directed to the selected X2 core inside the DynamIQ™ cluster. An example of a debug target is a development system with a test chip or a silicon part with a X2 core.

## Debug unit

Helps debugging software that is running on the core:

- DSU-110 and external hardware based around the core.
- Operating systems.
- Application software.

With the debug unit, you can:

- Stop program execution.
- Examine and alter process and coprocessor state.
- Examine and alter memory and the state of the input or output peripherals.
- Restart the *processing element* (PE).

For self-hosted debug, the debug target runs debug monitor software that runs on the core in the DynamIQ™ cluster. This way, it does not require expensive interface hardware to connect a second host computer.

## 16.2 Debug register interfaces

The Cortex®-X2 core implements the Arm®v9.0-A Debug architecture. It also supports the Arm®v8.4-A Debug architecture and Arm®v8.3-A Debug over powerdown.

The Debug architecture defines a set of Debug registers. The Debug register interfaces provide access to these registers either from software running on the core or from an external debugger. See *Debug* in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual* for more information.

### Related information

[5.6 Debug over powerdown](#) on page 47

### 16.2.1 Core interfaces

System register access allows the Cortex®-X2 core to access certain debug registers directly. The debug register interfaces provide access to these registers either from software running on the core or from an external debugger.

Access to the debug registers is partitioned as follows:

## Debug

This function is both system register based and memory-mapped. You can access the debug register map using the APB slave port that connects into the DebugBlock of the *DynamiQ™ Shared Unit-110* (DSU-110).

## Performance monitoring

This function is system register based and memory-mapped. You can access the performance monitor registers using the APB slave port that connects into the DebugBlock of the DSU.

## Trace

This function is system register based and memory-mapped. You can access the trace unit registers using the APB slave port that connects into the DebugBlock of the DSU.

## ELA registers

This function is memory-mapped. You can access the *Embedded Logic Analyzer* (ELA) registers using the APB slave port that connects into the DebugBlock of the DSU.

For information on the APB slave port interface, see *Interfaces* in the *Arm® DynamiQ™ Shared Unit-110 Technical Reference Manual*.

## 16.2.2 Effects of resets on debug registers

The **complexporeset\_n** and **complexreset\_n** signals of the core affect the debug registers.

**complexporeset\_n** maps to a Cold reset that covers reset of the core logic and the integrated debug functionality. This signal initializes the core logic, including the trace unit, breakpoint, watchpoint logic, performance monitor, and debug logic.

**complexreset\_n** maps to a Warm reset that covers reset of the core logic. This signal resets some of the debug and performance monitor logic.

## 16.2.3 External access permissions to Debug registers

External access permission to the Debug registers is subject to the conditions at the time of the access.

The following table shows the core response to accesses through the external debug interface.

**Table 16-1: External access conditions to registers**

Name	Condition	Description
Off	EDPRSR.PU = 1	Because Armv8.3-DoPD, Debug over PowerDown, is implemented, access to this field is <i>Read-As-One</i> (RAO). When the core power domain is in a powerup state, the Debug registers in the core power domain can be accessed. When the core power domain is OFF, accesses to the Debug registers in the core power domain, including EDPRSR, return an error.
OSLK	OSLSR_EL1.OSLK = 1	OS Lock is locked.
EDAD	AllowExternalDebugAccess() == FALSE	External debug access is disabled. If an error is returned because of an EDAD condition code, and this is the highest priority error condition, then EDPRSR.SDAD is set to 1. Otherwise, SDAD is unchanged.

Name	Condition	Description
Default	-	This is normal access, none of the conditions apply.

## 16.2.4 Breakpoints and watchpoints

The Cortex®-X2 core supports six breakpoints, four watchpoints, and a standard *Debug Communications Channel* (DCC).

A breakpoint consists of a breakpoint control register and a breakpoint value register. These two registers are referred to as a *Breakpoint Register Pair* (BRP). Four of the breakpoints (BRP 0-3) match only to the *Virtual Address* (VA) and the other two (BRP 4 and 5) match against either the VA or context ID, or the *Virtual Machine ID* (VMID).

You can use watchpoints to stop your target when a specific memory address is accessed by your program. All the watchpoints can be linked to two breakpoints (BRP 4 and 5) to enable a memory request to be trapped in a given process context.

## 16.3 Debug events

A debug event can be either a software debug event or a Halting debug event.

The Cortex®-X2 core responds to a debug event in one of the following ways:

- It ignores the debug event
- It takes a debug exception
- It enters debug state

In the Cortex®-X2 core, watchpoint debug events are always synchronous. Memory hint instructions and cache clean operations, except `dc zva`, and `dc ivac` do not generate watchpoint debug events. Store exclusive instructions generate a watchpoint debug event even when the check for the control of exclusive monitor fails. Atomic `cas` instructions generate a watchpoint debug event even when the compare operation fails.

A Cold reset sets the Debug OS Lock. For the debug events and debug register accesses to operate normally, the Debug OS Lock must be cleared.

## 16.4 Debug memory map and debug signals

The debug memory map and debug signals are handled at the DynamIQ™-110 cluster level.

See *Debug* and *ROM tables* in the *Arm® DynamIQ™ Shared Unit-110 Technical Reference Manual*.

## 16.5 ROM table

The Cortex®-X2 core includes a ROM table that contains a list of components in the system. Debuggers can use the ROM table to determine which CoreSight components are implemented.

The ROM table is a CoreSight debug related component that aids system debug along with CoreSight SoC and is for the Cortex®-X2 core. There is one ROM table for each core and ROM tables comply with the *Arm® CoreSight™ Architecture Specification v3.0*.

The *DynamiQ™ Shared Unit-110* (DSU-110) has its own ROM tables, one for the cluster and one for the DebugBlock, and has entry points in the cluster ROM table for the ROM tables belonging to each core. See *ROM tables* in the *Arm® DynamiQ™ Shared Unit-110 Technical Reference Manual* for more information.

The Cortex®-X2 core ROM table includes the following entries:

**Table 16-2: Core ROM table**

Offset	Name	Description
0x0000	ROMENTRY0	Core debug
0x0004	ROMENTRY1	Core PMU
0x0008	ROMENTRY2	Core Trace unit
0x000C	ROMENTRY3	Optional ELA

### Related information

[16.9 CoreROM register summary](#) on page 104

## 16.6 CoreSight component identification

Each component associated with the Cortex®-X2 core has a unique set of CoreSight ID values.

**Table 16-3: Cortex®-X2 CoreSight component identification**

Component	Peripheral ID	Component ID	DevType	DevArch	Revision
Trace Unit	0x04002BBD48	0xB105900D	0x00000013	0x47705a13	r2p1
PMU	0x04002BBD48	0xB105900D	0x00000016	0x47702a16	r2p1
DBG	0x04002BBD48	0xB105900D	0x00000015	0x47709a15	r2p1
ROM Table	0x04002BBD48	0xB105900D	0x00000000	0x47700af7	r2p1

For details on the CoreSight component identification for the Cortex®-X2 core ELA, see the *Arm® CoreSight™ ELA-600 Embedded Logic Analyzer Technical Reference Manual*.

## 16.7 AArch64 Debug register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Debug registers in the core. Individual register descriptions provide detailed information.

**Table 16-4: Debug register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
IMP_IDATA0_EL3	3	C15	6	C0	0	See individual bit resets.	64-bit	Instruction Register 0
IMP_IDATA1_EL3	3	C15	6	C0	1	See individual bit resets.	64-bit	Instruction Register 0
IMP_IDATA2_EL3	3	C15	6	C0	2	See individual bit resets.	64-bit	Instruction Register 0
IMP_DDATA0_EL3	3	C15	6	C1	0	See individual bit resets.	64-bit	Data Register 0
IMP_DDATA1_EL3	3	C15	6	C1	1	See individual bit resets.	64-bit	Data Register 1
IMP_DDATA2_EL3	3	C15	6	C1	2	See individual bit resets.	64-bit	Data Register 2

## 16.8 Debug register summary

The summary table provides an overview of all Debug registers in the core. Individual register descriptions provide detailed information.

**Table 16-5: Debug register summary**

Offset	Name	Reset	Width	Description
0x090	EDRCR	See individual bit resets.	32-bit	External Debug Reserve Control Register
0x094	EDACR	0x0	32-bit	External Debug Auxiliary Control Register
0x310	EDPRCR	See individual bit resets.	32-bit	External Debug Power/Reset Control Register
0xD00	MIDR_EL1	See individual bit resets.	32-bit	Main ID Register
0xD20	EDPFR	See individual bit resets.	64-bit	External Debug Processor Feature Register
0xD28	EDDFR	See individual bit resets.	64-bit	External Debug Feature Register
0xFBC	EDDEVARCH	See individual bit resets.	32-bit	External Debug Device Architecture register
0xFC0	EDDEVID2	0x0	32-bit	External Debug Device ID register 2
0xFC4	EDDEVID1	See individual bit resets.	32-bit	External Debug Device ID register 1
0xFC8	EDDEVID	See individual bit resets.	32-bit	External Debug Device ID register 0
0xFCC	EDDEVTYPE	See individual bit resets.	32-bit	External Debug Device Type register
0xFD0	EDPIDR4	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 4
0xFE0	EDPIDR0	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 0
0xFE4	EDPIDR1	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 1
0xFE8	EDPIDR2	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 2
0xFEC	EDPIDR3	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 3
0xFF0	EDCIDR0	See individual bit resets.	32-bit	External Debug Component Identification Register 0
0xFF4	EDCIDR1	See individual bit resets.	32-bit	External Debug Component Identification Register 1
0xFF8	EDCIDR2	See individual bit resets.	32-bit	External Debug Component Identification Register 2
0xFFC	EDCIDR3	See individual bit resets.	32-bit	External Debug Component Identification Register 3

## 16.9 CoreROM register summary

The summary table provides an overview of all CoreROM registers in the core. Individual register descriptions provide detailed information.

**Table 16-6: CoreROM register summary**

Offset	Name	Reset	Width	Description
0x000	COREROM_ROMENTRY0	See individual bit resets.	32-bit	Core ROM table Entry 0
0x004	COREROM_ROMENTRY1	See individual bit resets.	32-bit	Core ROM table Entry 1
0x008	COREROM_ROMENTRY2	See individual bit resets.	32-bit	Core ROM table Entry 2
0x00C	COREROM_ROMENTRY3	See individual bit resets.	32-bit	Core ROM table Entry 3
0xFB8	COREROM_AUTHSTATUS	See individual bit resets.	32-bit	Core ROM table Authentication Status Register
0xFBC	COREROM_DEVARCH	See individual bit resets.	32-bit	Core ROM table Device Architecture Register
0xFCC	COREROM_DEVTYPE	See individual bit resets.	32-bit	Core ROM table Device Type Register
0xFD0	COREROM_PIDR4	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 4
0xFE0	COREROM_PIDR0	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 0
0xFE4	COREROM_PIDR1	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 1
0xFE8	COREROM_PIDR2	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 2
0xFEC	COREROM_PIDR3	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 3
0xFF0	COREROM_CIDR0	See individual bit resets.	32-bit	Core ROM table Component Identification Register 0
0xFF4	COREROM_CIDR1	See individual bit resets.	32-bit	Core ROM table Component Identification Register 1
0xFF8	COREROM_CIDR2	See individual bit resets.	32-bit	Core ROM table Component Identification Register 2
0xFFC	COREROM_CIDR3	See individual bit resets.	32-bit	Core ROM table Component Identification Register 3



# 17 Performance Monitors Extension support

The Cortex®-X2 core implements the Performance Monitors Extension, including Arm®v8.4-A and Arm®v8.5-A performance monitoring features.

The Cortex®-X2 core *Performance Monitoring Unit* (PMU):

- Collects events through an event interface from other units in the design. These events are used as triggers for event counters.
- Supports cycle counters through the Performance Monitors Control Register.
- Implements PMU snapshots for context samples.
- Provides six or 20 PMU 64-bit counters that count any of the events available in the core. The absolute counts that are recorded might vary because of pipeline effects. This variation has negligible effect except in cases where the counters are enabled for a very short time.

You can program the PMU using either the System registers or the external Debug APB interface.

## 17.1 Performance monitors events

The Cortex®-X2 *Performance Monitoring Unit* (PMU) collects events from other units in the design and uses numbers to reference these events.

The following table lists the Cortex®-X2 performance monitors events. Event reference numbers that are not listed are reserved.



Unless otherwise indicated, each of these events can be exported to the Trace unit and selected in accordance with the Arm® *Embedded Trace Extension*.

**Table 17-1: Performance monitors Events**

Event number	Mnemonic	Description
0x0	SW_INCR	Software increment  This event counts any instruction architecturally executed (condition code check pass).
0x1	L1I_CACHE_REFILL	L1 instruction cache refill  This event counts any instruction fetch which misses in the cache.  The following instructions are not counted: <ul style="list-style-type: none"> <li>• Cache maintenance instructions</li> <li>• Non-cacheable accesses</li> </ul>

Event number	Mnemonic	Description
0x2	L1I_TLB_REFILL	<p>L1 instruction TLB refill</p> <p>This event counts any refill of the L1 instruction TLB from the <i>MMU Translation Cache</i> (MMUTC). This includes refills that result in a translation fault.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>• TLB maintenance instructions</li> </ul> <p>This event counts regardless of whether the MMU is enabled.</p>
0x3	L1D_CACHE_REFILL	<p>L1 data cache refill</p> <p>This event counts any load or store operation or translation table walk access which causes data to be read from outside the L1, including accesses which do not allocate into L1.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>• Cache maintenance instructions and prefetches</li> <li>• Stores of an entire cache line, even if they make a coherency request outside the L1</li> <li>• Partial cache line writes which do not allocate into the L1 cache.</li> <li>• Non-cacheable accesses</li> </ul> <p>This event counts the sum of L1D_CACHE_REFILL_RD and L1D_CACHE_REFILL_WR.</p>
0x4	L1D_CACHE	<p>L1 data cache access</p> <p>This event counts any load or store operation or translation table walk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL event causes this event to count.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>• Cache maintenance instructions and prefetches</li> <li>• Non-cacheable accesses</li> </ul> <p>This event counts the sum of L1D_CACHE_RD and L1D_CACHE_WR.</p>
0x5	L1D_TLB_REFILL	<p>L1 data TLB refill</p> <p>This event counts any refill of the data L1 TLB from the MMUTC. This includes refills that result in a translation fault. TLB maintenance instructions are not counted.</p> <p>This event counts regardless of whether the MMU is enabled.</p>
0x8	INST_RETIRED	<p>Instruction architecturally executed.</p> <p>This event counts all retired instructions, including those that fail their condition check.</p>
0x9	EXC_TAKEN	Exception taken
0x0A	EXC_RETURN	Instruction architecturally executed, condition code check pass, exception return
0x0B	CID_WRITE_RETIRED	<p>Instruction architecturally executed, condition code check pass, write to CONTEXTIDR</p> <p>This event only counts writes to CONTEXTIDR_EL1.</p> <p>Writes to CONTEXTIDR_EL12 and CONTEXTIDR_EL2 are not counted.</p>

Event number	Mnemonic	Description
0x10	BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed  This event counts any predictable branch instruction which is mispredicted either because of dynamic misprediction or because the MMU is off and the branches are statically predicted not taken.
0x11	CPU_CYCLES	Cycle
0x12	BR_PRED	Predictable branch speculatively executed.  This event counts all predictable branches.
0x13	MEM_ACCESS	Data memory access.  This event counts memory accesses due to load or store instructions.  The following instructions are not counted: <ul style="list-style-type: none"> <li>• Instruction fetches</li> <li>• Cache maintenance instructions</li> <li>• Translation table walks or prefetches</li> </ul> This event counts the sum of MEM_ACCESS_RD and MEM_ACCESS_WR.
0x14	L1I_CACHE	L1 instruction cache access or <i>Macro-op</i> (MOP) cache access.  This event counts any instruction fetch which accesses the L1 instruction cache or MOP cache.  The following instructions are not counted: <ul style="list-style-type: none"> <li>• Cache maintenance instructions</li> <li>• Non-cacheable accesses</li> </ul>
0x15	L1D_CACHE_WB	L1 data cache Write-Back  This event counts any write-back of data from the L1 data cache to L2 or L3. This counts both victim line evictions and snoops, including cache maintenance operations.  The following instructions are not counted: <ul style="list-style-type: none"> <li>• Invalidations which do not result in data being transferred out of the L1</li> <li>• Full-line writes which write to L2 without writing L1, such as write streaming mode</li> </ul>
0x16	L2D_CACHE	L2 cache access  This event counts any transaction from L1 which looks up in the L2 cache, and any write-back from the L1 to the L2.  Snoops from outside the core and cache maintenance operations are not counted.
0x17	L2D_CACHE_REFILL	L2 cache refill  This event counts any cacheable transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 are not counted.

Event number	Mnemonic	Description
0x18	L2D_CACHE_WB	<p>L2 cache write-back</p> <p>This event counts any write-back of data from the L2 cache to outside the core. This includes snoops to the L2 which return data, regardless of whether they cause an invalidation.</p> <p>Invalidation from the L2 which do not write data outside of the core and snoops which return data from the L1 are not counted.</p>
0x19	BUS_ACCESS	<p>Bus access</p> <p>This event counts for every beat of data transferred over the data channels between the core and the <i>Snoop Control Unit</i> (SCU). If both read and write data beats are transferred on a given cycle, this event is counted twice on that cycle.</p> <p>This event counts the sum of BUS_ACCESS_RD, BUS_ACCESS_WR, and any snoop data responses.</p>
0x1A	MEMORY_ERROR	<p>Local memory error</p> <p>This event counts any correctable or uncorrectable memory error (ECC or parity) in the protected core RAMs.</p>
0x1B	INST_SPEC	Operation speculatively executed
0x1C	TTBR_WRITE_RETIRED	<p>Instruction architecturally executed, condition code check pass, write to TTBR</p> <p>This event only counts writes to TTBR0_EL1/TTBR1_EL1.</p> <p>Accesses to TTBR0_EL12/TTBR1_EL12 or TTBR0_EL2/TTBR1_EL2 are not counted.</p>
0x1D	BUS_MASTER_CYCLE	<p>Bus cycles</p> <p>This event duplicates CPU_CYCLES.</p>
0x1E	COUNTER_OVERFLOW	For odd-numbered counters, this event increments the count by one for each overflow of the preceding even-numbered counter. For even-numbered counters, there is no increment.
0x20	CACHE_ALLOCATE	<p>L2 data cache allocation without refill.</p> <p>This event counts any full cache line write into the L2 cache which does not cause a linefill, including write-backs from L1 to L2 and full-line writes which do not allocate into L1.</p>
0x21	BR_RETIRED	<p>Instruction architecturally executed, branch</p> <p>This event counts all branches, taken or not. This excludes exception entries, debug entries and CCFAIL branches.</p>
0x22	BR_MIS_PRED_RETIRED	<p>Instruction architecturally executed, mispredicted branch</p> <p>This event counts any branch counted by BR_RETIRED which is not correctly predicted and causes a pipeline flush.</p>
0x23	STALL_FRONTEND	<p>No operation issued because of the frontend</p> <p>The counter counts on any cycle when there are no fetched instructions available to dispatch.</p>

Event number	Mnemonic	Description
0x24	STALL_BACKEND	No operation issued because of the backend  The counter counts on any cycle fetched instructions are not dispatched due to resource constraints.
0x25	L1D_TLB	Level 1 data TLB access  This event counts any load or store operation which accesses the data L1 TLB. If both a load and a store are executed on a cycle, this event counts twice. This event counts regardless of whether the MMU is enabled.
0x26	L1I_TLB	Level 1 instruction TLB access  This event counts any instruction fetch which accesses the instruction L1 TLB.  This event counts regardless of whether the MMU is enabled.
0x29	L3D_CACHE_ALLOCATE	Attributable L3 cache allocation without refill  This event counts any full cache line write into the L3 cache which does not cause a linefill, including write-backs from L2 to L3 and full-line writes which do not allocate into L2.
0x2A	L3D_CACHE_REFILL	Attributable L3 cache refill  This event counts for any cacheable read transaction returning data from the SCU for which the data source was outside the cluster.  Transactions such as ReadUnique are counted as read transactions, even though they can be generated by store instructions.
0x2B	L3D_CACHE	Attributable L3 cache access  This event counts for any cacheable read transaction returning data from the SCU, or for any cacheable write to the SCU.
0x2D	L2TLB_REFILL	Attributable L2 TLB refill  This event counts on any refill of the MMUTC, caused by either an instruction or data access.  This event does not count if the MMU is disabled.
0x2F	L2TLB_REQ	Attributable L2 TLB access  This event counts on any access to the MMUTC (caused by a refill of any of the L1 TLBs).  This event does not count if the MMU is disabled.
0x31	REMOTE_ACCESS	Access to another socket in a multi-socket system
0x34	DTLB_WLK	Access to data TLB that caused a page table walk  This event counts on any data access which causes L2D_TLB_REFILL to count.
0x35	ITLB_WLK	Access to instruction TLB that caused a translation table walk.  This event counts on any instruction access which causes L2D_TLB_REFILL to count.

Event number	Mnemonic	Description
0x36	LL_CACHE_RD	<p>Last level cache access, read</p> <p>If CPUECTLR.EXTLLC is set, then this event counts any cacheable read transaction which returns a data source of interconnect cache.</p> <p>If CPUECTLR.EXTLLC is not set, then this event is a duplicate of the L*D_CACHE_RD event corresponding to the last level of cache implemented L2D_CACHE_RD if only one is implemented, or L1D_CACHE_RD if neither is implemented.</p>
0x37	LL_CACHE_MISS_RD	<p>Last level cache miss, read</p> <p>If CPUECTLR.EXTLLC is set, then this event counts any cacheable read transaction which returns a data source of DRAM, remote, or inter-cluster peer.</p> <p>If CPUECTLR.EXTLLC is not set, then this event is a duplicate of the L*D_CACHE_REFILL_RD event corresponding to the last level of cache implemented L2D_CACHE_REFILL_RD if only one is implemented, or L1D_CACHE_REFILL_RD if neither is implemented.</p>
0x39	L1D_CACHE_LMISS_RD	Level 1 data cache long-latency miss
0x3A	OP_RETIRED	Micro-operation architecturally executed
0x3B	OP_SPEC	Micro-operation speculatively executed
0x3C	STALL	No operation sent for execution
0x3D	STALL_SLOT_BACKEND	No operation sent for execution on a slot due to the backend
0x3E	STALL_SLOT_FRONTEND	No operation sent for execution on a slot due to the frontend
0x3F	STALL_SLOT	No operation sent for execution on a slot
0x40	L1D_CACHE_RD	<p>L1 data cache access, read</p> <p>This event counts any load operation or page table walk access which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_RD event causes this event to count.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>Cache maintenance instructions and prefetches</li> <li>Non-cacheable accesses</li> </ul>
0x41	L1D_CACHE_WR	<p>L1 data cache access, write</p> <p>This event counts any store operation which looks up in the L1 data cache. In particular, any access which could count the L1D_CACHE_REFILL_WR event causes this event to count.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>Cache maintenance instructions and prefetches</li> <li>Non-cacheable accesses</li> </ul>

Event number	Mnemonic	Description
0x42	L1D_CACHE_REFILL_RD	<p>L1 data cache refill, read</p> <p>This event counts any load operation or page table walk access which causes data to be read from outside the L1, including accesses which do not allocate into L1.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>Cache maintenance instructions and prefetches</li> <li>Non-cacheable accesses</li> </ul>
0x43	L1D_CACHE_REFILL_WR	<p>L1 data cache refill, write</p> <p>This event counts any store operation which causes data to be read from outside the L1, including accesses which do not allocate into L1.</p> <p>The following instructions are not counted:</p> <ul style="list-style-type: none"> <li>Cache maintenance instructions and prefetches</li> <li>Stores of an entire cache line, even if they make a coherency request outside the L1</li> <li>Partial cache line writes which do not allocate into the L1 cache.</li> <li>Non-cacheable accesses</li> </ul>
0x44	L1D_CACHE_REFILL_INNER	<p>L1 data cache refill, inner</p> <p>This event counts any L1 data cache linefill (as counted by L1D_CACHE_REFILL) which hits in the L2 cache, system L3 cache, or another core in the cluster.</p>
0x45	L1D_CACHE_REFILL_OUTER	<p>L1 data cache refill, outer</p> <p>This event counts any L1 data cache linefill (as counted by L1D_CACHE_REFILL) which does not hit in the L2 cache, system L3 cache, or another core in the cluster, and instead obtains data from outside the cluster.</p>
0x46	L1D_CACHE_WB_VICTIM	L1 data cache write-back, victim
0x47	L1D_CACHE_WB_CLEAN	L1 data cache write-back cleaning and coherency
0x48	L1D_CACHE_INVALID	L1 data cache invalidate
0x4C	L1D_TLB_REFILL_RD	L1 data TLB refill, read
0x4D	L1D_TLB_REFILL_WR	L1 data TLB refill, write
0x4E	L1D_TLB_RD	L1 data TLB access, read
0x4F	L1D_TLB_WR	L1 data TLB access, write
0x50	CACHE_ACCESS_RD	<p>L2 cache access, read</p> <p>This event counts any read transaction from L1 which looks up in the L2 cache.</p> <p>Snoops from outside the core are not counted.</p>
0x51	CACHE_ACCESS_WR	<p>L2 cache access, write</p> <p>This event counts any write transaction from L1 which looks up in the L2 cache or any write-back from L1 which allocates into the L2 cache.</p> <p>Snoops from outside the core are not counted.</p>

Event number	Mnemonic	Description
0x52	CACHE_RD_REFILL	L2 cache refill, read  This event counts any cacheable read transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 should not be counted. Transactions such as ReadUnique are counted as read transactions, even though they can be generated by store instructions.
0x53	CACHE_WR_REFILL	L2 cache refill, write  This event counts any write transaction from L1 which causes data to be read from outside the core. L2 refills caused by stashes into L2 should not be counted.  Transactions such as ReadUnique are not counted as write transactions.
0x56	CACHE_WRITEBACK_VICTIM	L2 cache write-back, victim
0x57	CACHE_WRITEBACK_CLEAN_COH	L2 cache write-back, cleaning and coherency
0x58	L2CACHE_INV	L2 cache invalidate
0x5C	L2TLB_RD_REFILL	L2 TLB refill, read
0x5D	L2TLB_WR_REFILL	L2 TLB refill, write
0x5E	L2TLB_RD_REQ	L2 TLB access, read
0x5F	L2TLB_WR_REQ	L2 TLB access, write
0x60	BUS_ACCESS_RD	Bus access read  This event counts for every beat of data transferred over the read data channel between the core and the SCU.
0x61	BUS_ACCESS_WR	Bus access write  This event counts for every beat of data transferred over the write data channel between the core and the SCU.
0x66	MEM_ACCESS_RD	Data memory access, read  This event counts memory accesses due to load instructions.  The following instructions are not counted: <ul style="list-style-type: none"> <li>• Instruction fetches</li> <li>• Cache maintenance instructions</li> <li>• Translation table walks</li> <li>• Prefetches</li> </ul>



Event number	Mnemonic	Description
0x67	MEM_ACCESS_WR	Data memory access, write  This event counts memory accesses due to store instructions.  The following instructions are not counted: <ul style="list-style-type: none"> <li>• Instruction fetches.</li> <li>• Cache maintenance instructions</li> <li>• Translation table walks</li> <li>• Prefetches</li> </ul>
0x68	UNALIGNED_LD_SPEC	Unaligned access, read
0x69	UNALIGNED_ST_SPEC	Unaligned access, write
0x6A	UNALIGNED_LDST_SPEC	Unaligned access
0x6C	LDREX_SPEC	Exclusive operation speculatively executed, LDREX or LDX
0x6D	STREX_PASS_SPEC	Exclusive operation speculatively executed, STREX or STX pass
0x6E	STREX_FAIL_SPEC	Exclusive operation speculatively executed, STREX or STX fail
0x6F	STREX_SPEC	Exclusive operation speculatively executed, STREX or STX
0x70	LD_SPEC	Operation speculatively executed, load
0x71	ST_SPEC	Operation speculatively executed, store
0x73	DP_SPEC	Operation speculatively executed, integer data-processing
0x74	ASE_SPEC	Operation speculatively executed, Advanced SIMD instruction
0x75	VFP_SPEC	Operation speculatively executed, floating-point instruction
0x76	PC_WRITE_SPEC	Operation speculatively executed, software change of the PC
0x77	CRYPTO_SPEC	Operation speculatively executed, Cryptographic instruction
0x78	BR_IMMED_SPEC	Branch speculatively executed, immediate branch
0x79	BR_RETURN_SPEC	Branch speculatively executed, procedure return
0x7A	BR_INDIRECT_SPEC	Branch speculatively executed, indirect branch
0x7C	ISB_SPEC	Barrier speculatively executed, ISB
0x7D	DSB_SPEC	Barrier speculatively executed, DSB
0x7E	DMB_SPEC	Barrier speculatively executed, DMB
0x81	EXC_UNDEF	Counts the number of undefined exceptions taken locally
0x82	EXC_SVC	Exception taken locally, Supervisor Call
0x83	EXC_PABORT	Exception taken locally, Instruction Abort
0x84	EXC_DABORT	Exception taken locally, Data Abort and SError
0x86	EXC_IRQ	Exception taken locally, IRQ
0x87	EXC_FIQ	Exception taken locally, FIQ
0x88	EXC_SMC	Exception taken locally, Secure Monitor Call
0x8A	EXC_HVC	Exception taken locally, Hypervisor Call
0x8B	EXC_TRAP_PABORT	Exception taken, Instruction Abort not taken locally
0x8C	EXC_TRAP_DABORT	Exception taken, Data Abort or SError not taken locally

Event number	Mnemonic	Description
0x8D	EXC_TRAP_OTHER	Exception taken, Other traps not taken locally
0x8E	EXC_TRAP_IRQ	Exception taken, IRQ not taken locally
0x8F	EXC_TRAP_FIQ	Exception taken, FIQ not taken locally
0x90	RC_LD_SPEC	Release consistency operation speculatively executed, load-acquire
0x91	RC_ST_SPEC	Release consistency operation speculatively executed, store-release
0xA0	L3_CACHE_RD	L3 cache read
0x4004	CNT_CYCLES	Constant frequency cycles
0x4005	STALL_BACKEND_MEM	No operation sent due to the backend and memory stalls
0x4006	L1I_CACHE_LMISS	L1 instruction cache long latency miss
0x4009	L2D_CACHE_LMISS_RD	L2 cache long latency miss
0x400B	L3D_CACHE_LMISS_RD	L3 cache long latency miss
0x400C	TRB_WRAP	Trace buffer current write pointer wrapped
0x4010	TRCEXTOUT0	PE Trace Unit external output 0
0x4011	TRCEXTOUT1	PE Trace Unit external output 1
0x4012	TRCEXTOUT2	PE Trace Unit external output 2
0x4013	TRCEXTOUT3	PE Trace Unit external output 3
0x4018	CTI_TRIGOUT4	Cross-trigger Interface output trigger 4
0x4019	CTI_TRIGOUT5	Cross-trigger Interface output trigger 5
0x401A	CTI_TRIGOUT6	Cross-trigger Interface output trigger 6
0x401B	CTI_TRIGOUT7	Cross-trigger Interface output trigger 7
0x4020	LDST_ALIGN_LAT	Access with additional latency from alignment
0x4021	LD_ALIGN_LAT	Load with additional latency from alignment
0x4022	ST_ALIGN_LAT	Store with additional latency from alignment
0x4024	MEM_ACCESS_CHECKED	Checked data memory access
0x4025	MEM_ACCESS_RD_CHECKED	Checked data memory access, read
0x4026	MEM_ACCESS_WR_CHECKED	Checked data memory access, write
0x8005	ASE_INST_SPEC	Advanced SIMD operations speculatively executed
0x8006	SVE_INST_SPEC	SVE operations speculatively executed
0x8014	FP_HP_SPEC	Half-precision floating-point operation speculatively executed
0x8018	FP_SP_SPEC	Single-precision floating-point operation speculatively executed
0x801C	FP_DP_SPEC	Double-precision floating-point operation speculatively executed
0x8074	SVE_PRED_SPEC	SVE predicated operations speculatively executed
0x8075	SVE_PRED_EMPTY_SPEC	SVE predicated operations with no active predicates speculatively executed
0x8076	SVE_PRED_FULL_SPEC	SVE predicated operations speculatively executed with all active predicates
0x8077	SVE_PRED_PARTIAL_SPEC	SVE predicated operations speculatively executed with partially active predicates
0x8079	SVE_PRED_NOT_FULL_SPEC	SVE predicated operations speculatively executed with a Governing predicate in which at least one element is FALSE
0x80BC	SVE_LDFF_SPEC	SVE First-fault load operations speculatively executed
0x80BD	SVE_LDFF_FAULT_SPEC	SVE First-fault load operations speculatively executed which set FFR bit to 0
0x80C0	FP_SCALE_OPS_SPEC	Scalable floating-point element operations speculatively executed

Event number	Mnemonic	Description
0x80C1	FP_FIXED_OPS_SPEC	Non-scalable floating-point element operations speculatively executed
0x80E3	ASE_SVE_INT8_SPEC	Operation counted by ASE_SVE_INT_SPEC where the largest type is 8-bit integer
0x80E7	ASE_SVE_INT16_SPEC	Operation counted by ASE_SVE_INT_SPEC where the largest type is 16-bit integer
0x80EB	ASE_SVE_INT32_SPEC	Operation counted by ASE_SVE_INT_SPEC where the largest type is 32-bit integer
0x80EF	ASE_SVE_INT64_SPEC	Operation counted by ASE_SVE_INT_SPEC where the largest type is 64-bit integer

## 17.2 Performance monitors interrupts

The *Performance Monitoring Unit* (PMU) can be configured to generate an interrupt when one or more of the counters overflow.

When the PMU generates an interrupt, the **nPMUIRQ[n]** output is driven LOW.

## 17.3 External register access permissions

The Cortex®-X2 core supports access to the *Performance Monitoring Unit* (PMU) registers from the system register interface and a memory-mapped interface.

Access to a register depends on:

- Whether the core is powered up
- The state of the OS Lock
- The state of External Performance Monitors Access Disable

The behavior is specific to each register and is not described in this manual. For a detailed description of these features and their effects on the registers, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*. The register descriptions provided in this manual describe whether each register is read/write or read-only.

## 17.4 AArch64 Performance Monitors register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Performance Monitors registers in the core. Individual register descriptions provide detailed information.

**Table 17-2: Performance Monitors register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
PMMIR_EL1	3	C9	0	C14	6	See individual bit resets.	64-bit	Performance Monitors Machine Identification Register
PMCR_ELO	3	C9	3	C12	0	See individual bit resets.	64-bit	Performance Monitors Control Register

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
PMCEID0_ELO	3	C9	3	C12	6	See individual bit resets.	64-bit	Performance Monitors Common Event Identification register 0
PMCEID1_ELO	3	C9	3	C12	7	See individual bit resets.	64-bit	Performance Monitors Common Event Identification register 1

## 17.5 PMU register summary

The summary table provides an overview of all PMU registers in the core. Individual register descriptions provide detailed information.

**Table 17-3: PMU register summary**

Offset	Name	Reset	Width	Description
0x600	PMPCSSR	See individual bit resets.	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register
0x60C	PMCID2SSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL2 Sample Register
0x610	PMSSSR	0x1	32-bit	PMU Snapshot Status Register
0x618	PMCCNTSR	See individual bit resets.	64-bit	PMU Cycle Counter Snapshot Register
0x620	PMEVCNTSR0	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x628	PMEVCNTSR1	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x630	PMEVCNTSR2	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x638	PMEVCNTSR3	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x640	PMEVCNTSR4	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x648	PMEVCNTSR5	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x650	PMEVCNTSR6	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x658	PMEVCNTSR7	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x660	PMEVCNTSR8	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x668	PMEVCNTSR9	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x670	PMEVCNTSR10	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x678	PMEVCNTSR11	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x680	PMEVCNTSR12	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x688	PMEVCNTSR13	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x690	PMEVCNTSR14	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x698	PMEVCNTSR15	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6A0	PMEVCNTSR16	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6A8	PMEVCNTSR17	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6B0	PMEVCNTSR18	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6B8	PMEVCNTSR19	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6F0	PMSSCR	See individual bit resets.	32-bit	PMU Snapshot Capture Register
0xE00	PMCFGR	See individual bit resets.	32-bit	Performance Monitors Configuration Register
0xE04	PMCR_ELO	See individual bit resets.	32-bit	Performance Monitors Control Register

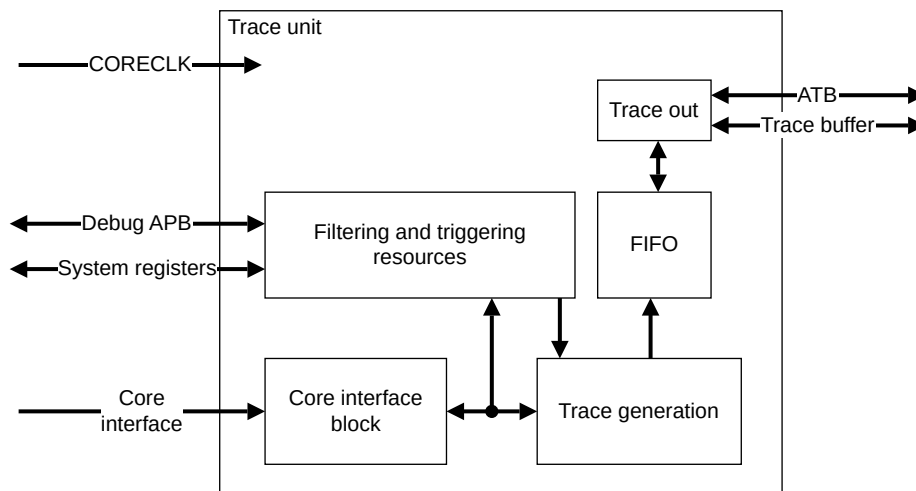
Offset	Name	Reset	Width	Description
0xE20	<a href="#">PMCEID0</a>	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 0
0xE24	<a href="#">PMCEID1</a>	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 1
0xE28	<a href="#">PMCEID2</a>	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 2
0xE2C	<a href="#">PMCEID3</a>	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 3
0xE40	<a href="#">PMMIR</a>	See individual bit resets.	32-bit	Performance Monitors Machine Identification Register
0xFBC	<a href="#">PMDEVARCH</a>	See individual bit resets.	32-bit	Performance Monitors Device Architecture register
0xFC8	<a href="#">PMDEVID</a>	See individual bit resets.	32-bit	Performance Monitors Device ID register
0xFCC	<a href="#">PMDEVTYPE</a>	See individual bit resets.	32-bit	Performance Monitors Device Type register
0xFD0	<a href="#">PMPIDR4</a>	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	<a href="#">PMPIDR0</a>	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	<a href="#">PMPIDR1</a>	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	<a href="#">PMPIDR2</a>	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	<a href="#">PMPIDR3</a>	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	<a href="#">PMCIDR0</a>	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 0
0xFF4	<a href="#">PMCIDR1</a>	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 1
0xFF8	<a href="#">PMCIDR2</a>	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 2
0xFFC	<a href="#">PMCIDR3</a>	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 3

# 18 Embedded Trace Extension support

The Cortex®-X2 core implements the *Embedded Trace Extension* (ETE). The trace unit performs real-time instruction flow tracing based on the ETE. The trace unit is a CoreSight component and is an integral part of the Arm real-time debug solution.

The following figure shows the main components of the trace unit:

**Figure 18-1: Trace unit components**



## Core interface

The core interface monitors and generates P0 elements that are essentially executed branches and exceptions traced in program order.

## Trace generation

The trace generation logic generates various trace packets based on P0 elements.

## Filtering and triggering resources

You can limit the amount of trace data that the trace unit generates by filtering. For example, you can limit trace generation to a certain address range. The trace unit supports other logic analyzer style filtering options. The trace unit can also generate a trigger that is a signal to the Trace Capture Device to stop capturing trace.

## FIFO

The trace unit generates trace in a highly compressed form. The *First In First Out* (FIFO) enables trace bursts to be flattened out. When the FIFO is full, the FIFO signals an overflow. The trace generation logic does not generate any new trace until the FIFO is emptied. This behavior causes a gap in the trace when viewed in the debugger.

## Trace out

Trace from the FIFO is output on the AMBA ATB interface or to the trace buffer.

See the *Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile* for more information.

## 18.1 Trace unit resources

Trace resources include counters, external input and output signals, and comparators.

The following table shows the trace unit resources, and indicates which of these resources the X2 core implements.

**Table 18-1: Trace unit resources implemented**

Description	Configuration
Number of resource selection pairs implemented	8
Number of external input selectors implemented	4
Number of <i>Embedded Trace Extension</i> (ETE) events	4
Number of counters implemented	2
Reduced function counter implemented	Not implemented
Number of sequencer states implemented	4
Number of Virtual Machine ID comparators implemented	1
Number of Context ID comparators implemented	1
Number of address comparator pairs implemented	4
Number of single-shot comparator controls	1
Number of core comparator inputs implemented	0
Data address comparisons implemented	Not implemented
Number of data value comparators implemented	0

See the *Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile* for more information.

## 18.2 Trace unit generation options

The Cortex®-X2 core trace unit implements a set of generation options.

The following table shows the trace generation options that are implemented in the Cortex®-X2 core trace unit.

**Table 18-2: Trace unit generation options implemented**

Description	Configuration
Instruction address size in bytes	8
Data address size in bytes	0, as the <i>Embedded Trace Extension</i> (ETE) does not implement data tracing
Data value size in bytes	0, as the ETE does not implement data tracing
Virtual Machine ID size in bytes	4
Context ID size in bytes	4
Support for conditional instruction tracing	Not implemented
Support for tracing of data	Not implemented
Support for tracing of load and store instructions as PO elements	Not implemented
Support for cycle counting in the instruction trace	Implemented
Support for branch broadcast tracing	Implemented
Number of events that are supported in the trace	4
Return stack support	Implemented
Tracing of SError exception support	Implemented
Instruction trace cycle counting minimum threshold	4
Size of Trace ID	7 bits
Synchronization period support	Read/write
Global timestamp size	64 bits
Number of cores available for tracing	1
ATB trigger support	Implemented
Low-power behavior override	Not implemented
Stall control support	Not implemented
Support for overflow avoidance	Not implemented
Support for using CONTEXTIDR_EL2 in <i>Virtual Machine Identifier</i> (VMID) comparator	Implemented

See the *Arm® Architecture Reference Manual Supplement Armv9*, for Armv9-A architecture profile for more information.

## 18.3 Reset the trace unit

The reset for the trace buffer is the same as a Cold reset for the core. When using the *TRace Buffer Extension* (TRBE), a Warm reset disables the trace buffer and therefore it is not possible to use the trace buffer to capture trace for a Warm reset.

If the trace unit is reset, then tracing stops until the trace unit is reprogrammed and re-enabled. However, if the core is reset using Warm reset, the last few instructions provided by the core before the reset might not be traced.



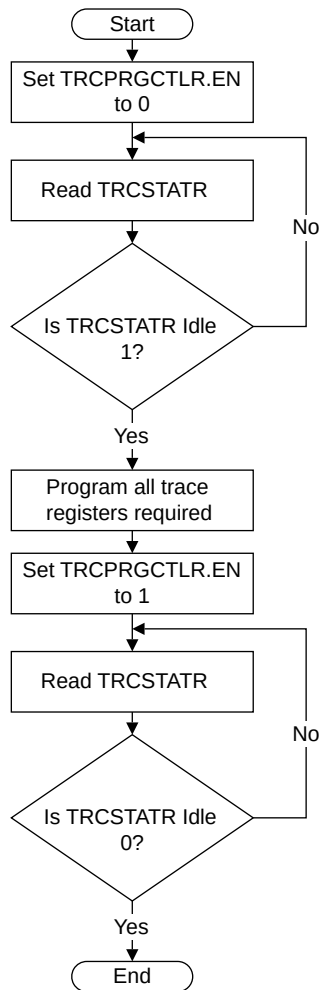
## 18.4 Program and read the trace unit registers

You program and read the trace unit registers using either the Debug APB interface or the System register interface.

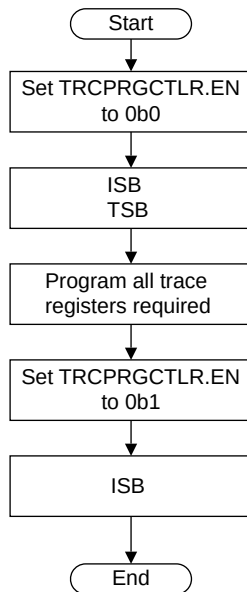
The core does not have to be in debug state when you program the trace unit registers. When you program the trace unit registers, you must enable all the changes at the same time. Otherwise, if you program the counter, it might start to count based on incorrect events before the correct setup is in place for the trigger condition. To disable the trace unit, use the TRCPRGCTLR.EN bit. See the *Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile* for more information about the following registers:

- Programming Control Register, TRCPRGCTLR
- Trace Status Register, TRCSTATR

The following figure shows the flow for programming trace unit registers using the Debug APB interface:

**Figure 18-2: Programming trace unit registers using the Debug APB interface**

The following figure shows the flow for programming trace unit registers using the System register interface:

**Figure 18-3: Programming trace registers using the System register interface**

## 18.5 Trace unit register interfaces

The Cortex®-X2 core supports an APB memory-mapped interface and a system register interface to trace unit registers.

Register accesses differ depending on the trace unit state. See the *Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile* for information on the behaviors and access mechanisms.

## 18.6 Interaction with the Performance Monitoring Unit and Debug

The trace unit interacts with the *Performance Monitoring Unit* (PMU) and it can access the PMU events.

### Interaction with the PMU

The Cortex®-X2 core includes a PMU that enables events, such as cache misses and executed instructions, to be counted over time.

The PMU and trace unit function together.

### Use of PMU events by the trace unit

The PMU architectural events are available to the trace unit through the extended input facility.

The trace unit uses four extended external input selectors to access the PMU events. Each selector can independently select one of the PMU events, that are then active for the cycles where the relevant events occur. These selected events can then be accessed by any of the event registers within the trace unit.

### Related information

[17 Performance Monitors Extension support](#) on page 105

[17.1 Performance monitors events](#) on page 105

## 18.7 ETE events

The Cortex®-X2 trace unit collects events from other units in the design and uses numbers to reference these events.

Other than the events mentioned in [17.1 Performance monitors events](#) on page 105, the following events are also exported:

**Table 18-3: ETE events**

Event number	Event mnemonic	Description
0x400D	PMU_OVFS	PMU overflow, counters accessible to EL1 and EL0
0x400E	TRB_TRIG	Trace buffer Trigger Event
0x400F	PMU_HOVS	PMU overflow, counters reserved for use by EL2

## 18.8 AArch64 Trace register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Trace registers in the core. Individual register descriptions provide detailed information.

**Table 18-4: Trace register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
<a href="#">TRCIDR8</a>	2	C0	1	C0	6	See individual bit resets.	64-bit	ID Register 8
<a href="#">TRCIMSPECO</a>	2	C0	1	C0	7	See individual bit resets.	64-bit	IMP DEF Register 0
<a href="#">TRCIDR2</a>	2	C0	1	C10	7	See individual bit resets.	64-bit	ID Register 2
<a href="#">TRCIDR3</a>	2	C0	1	C11	7	See individual bit resets.	64-bit	ID Register 3
<a href="#">TRCIDR4</a>	2	C0	1	C12	7	See individual bit resets.	64-bit	ID Register 4
<a href="#">TRCIDR5</a>	2	C0	1	C13	7	See individual bit resets.	64-bit	ID Register 5
<a href="#">TRCIDR10</a>	2	C0	1	C2	6	0x0	64-bit	ID Register 10
<a href="#">TRCIDR11</a>	2	C0	1	C3	6	0x0	64-bit	ID Register 11
<a href="#">TRCIDR12</a>	2	C0	1	C4	6	0x0	64-bit	ID Register 12
<a href="#">TRCIDR13</a>	2	C0	1	C5	6	0x0	64-bit	ID Register 13
<a href="#">TRCIDR0</a>	2	C0	1	C8	7	See individual bit resets.	64-bit	ID Register 0

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
TRCIDR1	2	C0	1	C9	7	See individual bit resets.	64-bit	ID Register 1
TRCCIDCVRO	2	C3	1	C0	0	See individual bit resets.	64-bit	Context Identifier Comparator Value Registers <n>

## 18.9 ETE register summary

The summary table provides an overview of all ETE registers in the core. Individual register descriptions provide detailed information.

**Table 18-5: ETE register summary**

Offset	Name	Reset	Width	Description
0x018	TRCAUXCTLR	0x0	32-bit	Auxillary Control Register
0x180	TRCIDR8	See individual bit resets.	32-bit	ID Register 8
0x184	TRCIDR9	See individual bit resets.	32-bit	ID Register 9
0x188	TRCIDR10	See individual bit resets.	32-bit	ID Register 10
0x18C	TRCIDR11	See individual bit resets.	32-bit	ID Register 11
0x190	TRCIDR12	0x0	32-bit	ID Register 12
0x194	TRCIDR13	0x0	32-bit	ID Register 13
0x1C0	TRCIMSPEC0	See individual bit resets.	32-bit	IMP DEF Register 0
0x1E0	TRCIDR0	See individual bit resets.	32-bit	ID Register 0
0x1E4	TRCIDR1	See individual bit resets.	32-bit	ID Register 1
0x1E8	TRCIDR2	See individual bit resets.	32-bit	ID Register 2
0x1EC	TRCIDR3	See individual bit resets.	32-bit	ID Register 3
0x1F0	TRCIDR4	See individual bit resets.	32-bit	ID Register 4
0x1F4	TRCIDR5	See individual bit resets.	32-bit	ID Register 5
0x1F8	TRCIDR6	0x0	32-bit	ID Register 6
0x1FC	TRCIDR7	0x0	32-bit	ID Register 7
0xF00	TRCITCTRL	See individual bit resets.	32-bit	Integration Mode Control Register
0xFA0	TRCCLAIMSET	See individual bit resets.	32-bit	Claim Tag Set Register
0xFA4	TRCCLAIMCLR	See individual bit resets.	32-bit	Claim Tag Clear Register
0xFBC	TRCDEVARCH	See individual bit resets.	32-bit	Device Architecture Register
0xFC0	TRCDEVID2	0x0	32-bit	Device Configuration Register 2
0xFC4	TRCDEVID1	0x0	32-bit	Device Configuration Register 1
0xFC8	TRCDEVID	0x0	32-bit	Device Configuration Register
0xFCC	TRCDEVTYPE	See individual bit resets.	32-bit	Device Type Register
0xFD0	TRCPIDR4	See individual bit resets.	32-bit	Peripheral Identification Register 4
0xFD4	TRCPIDR5	0x0	32-bit	Peripheral Identification Register 5
0xFD8	TRCPIDR6	0x0	32-bit	Peripheral Identification Register 6
0xFDC	TRCPIDR7	0x0	32-bit	Peripheral Identification Register 7
0xFE0	TRCPIDR0	See individual bit resets.	32-bit	Peripheral Identification Register 0
0xFE4	TRCPIDR1	See individual bit resets.	32-bit	Peripheral Identification Register 1

Offset	Name	Reset	Width	Description
0xFE8	<a href="#">TRCPIDR2</a>	See individual bit resets.	32-bit	Peripheral Identification Register 2
0xFEC	<a href="#">TRCPIDR3</a>	See individual bit resets.	32-bit	Peripheral Identification Register 3
0xFF0	<a href="#">TRCCIDR0</a>	See individual bit resets.	32-bit	Component Identification Register 0
0xFF4	<a href="#">TRCCIDR1</a>	See individual bit resets.	32-bit	Component Identification Register 1
0xFF8	<a href="#">TRCCIDR2</a>	See individual bit resets.	32-bit	Component Identification Register 2
0xFFC	<a href="#">TRCCIDR3</a>	See individual bit resets.	32-bit	Component Identification Register 3

# 19 Trace Buffer Extension support

The Cortex®-X2 core implements the *TRace Buffer Extension* (TRBE). The TRBE writes the program flow trace generated by the trace unit directly to memory. The TRBE is programmed through System registers.

When enabled, the TRBE can:

- Accept trace data from the trace unit and write it to L2 memory.
- Discard trace data from the trace unit. In this case, the data is lost.
- Reject trace data from the trace unit. In this case, the trace unit retains data until the TRBE accepts it.

When disabled, the TRBE ignores trace data and the trace unit sends trace data to the AMBA® *Trace Bus* (ATB) interface.

## 19.1 Program and read the trace buffer registers

You can program and read the *TRace Buffer Extension* (TRBE) registers using the System register interface.

The core does not have to be in debug state when you program the TRBE registers. When you program the TRBE registers, you must enable all the changes at the same time. Otherwise, if you program the counter, it might start to count based on incorrect events before the correct setup is in place for the trigger condition. To disable the TRBE, use the TRBLIMITR\_EL1.E bit.

See the *Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile* for information on the TRBE register behaviors and access mechanisms.

## 19.2 Trace buffer register interface

The Cortex®-X2 core supports a System register interface to *TRace Buffer Extension* (TRBE) registers.

Register accesses differ depending on the TRBE state. See the *Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile* for information on the behaviors and access mechanisms.

## 20 Activity Monitors Extension support

The Cortex®-X2 core implements the Activity Monitors Extension to the Arm®v8.4-A architecture. Activity monitoring has features similar to performance monitoring features, but is intended for system management use whereas performance monitoring is aimed at user and debug applications.

The activity monitors provide useful information for system power management and persistent monitoring. The activity monitors are read-only in operation and their configuration is limited to the highest Exception level implemented.

The Cortex®-X2 core implements seven counters in two groups, each of which is a 64-bit counter that counts a fixed event. Group 0 has four counters 0-3, and Group 1 has three counters 0-2.

### 20.1 Activity monitors access

The Cortex®-X2 core supports access to activity monitors from the System register interface and supports read-only memory-mapped access using the Utility bus interface.

See the *Arm® Architecture Reference Manual Armv8, for A-profile architecture* for information on the memory mapping of these registers.

#### Access enable bit

The access enable bit `AMUSERENR_ELO.EN` controls access from EL0 to the activity monitors System registers.

The `CPTR_EL2.TAM` bit controls access from EL0 and EL1 to the activity monitors System registers. The `CPTR_EL3.TAM` bit controls access from EL0, EL1, and EL2 to the Activity Monitors Extension System registers. The `AMUSERENR_ELO.EN` bit is configurable at EL1, EL2, and EL3. All other controls, as well as the value of the counters, are configurable only at the highest implemented Exception level.

For a detailed description of access controls for the registers, see the *Arm® Architecture Reference Manual Armv8, for A-profile architecture*.

#### System register access

The activity monitors are accessible using the `MRS` and `MSR` instructions.

#### External memory-mapped access

Activity monitors can be memory-mapped accessed from the Utility bus interface. In this case, the activity monitors registers only provide read access to the Activity Monitor Event Counter Registers.

The base address for *Activity Monitoring Unit* (AMU) registers on the Utility bus interface is `0x<n>90000`, where *n* is the Cortex®-X2 core instance number in the DynamIQ™-110 cluster.

These registers are treated as RAZ/WI if either:



- The register is marked as Reserved.
- The register is accessed in the wrong Security state.

## 20.2 Activity monitors counters

The Cortex®-X2 core implements four activity monitors counters, 0-3, and three auxiliary counters, 10-12.

Each counter has the following characteristics:

- All events are counted in 64-bit wrapping counters that overflow when they wrap. There is no support for overflow status indication or interrupts.
- Any change in clock frequency, including when a `WFI` and `WFE` instruction stops the clock, can affect any counter.
- Events 0-3 and auxiliary events 10-12 are fixed, and the `AMEVTYPER0<n>_ELO` and `AMEVTYPER1<n>_ELO` evtCount bits are read-only.
- The activity monitor counters are reset to zero on a Cold reset of the power domain of the core. When the core is not in reset, activity monitoring is available.

## 20.3 Activity monitors events

Activity monitors events in the Cortex®-X2 core are either fixed or programmable, and they map to the activity monitors counters.

The following table shows the mapping of counters to fixed events.

**Table 20-1: Mapping of counters to fixed events**

Activity monitor counter <n>	Event	Event number	Description
AMEVCNTR00	CPU_CYCLES	0x0011	Core frequency cycles
AMEVCNTR01	CNT_CYCLES	0x4004	Constant frequency cycles
AMEVCNTR02	Instructions retired	0x0008	Instruction architecturally executed  This counter increments for every instruction that is executed architecturally, including instructions that fail their condition code check.
AMEVCNTR03	STALL_BACKEND_MEM	0x4005	Memory stall cycles  This counter counts cycles in which the core is unable to dispatch instructions from the front end to the back end due to a back end stall caused by a miss in the last level of cache within the core clock domain.
AMEVCNTR10	Reserved	0x0300	Reserved
AMEVCNTR11	Reserved	0x0301	Reserved
AMEVCNTR12	Reserved	0x0302	Reserved

## 20.4 AArch64 Activity Monitors register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Activity Monitors registers in the core. Individual register descriptions provide detailed information.

**Table 20-2: Activity Monitors register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
AMEVTYPER10_ELO	3	C13	3	C14	0	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_ELO	3	C13	3	C14	1	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_ELO	3	C13	3	C14	2	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMCFGR_ELO	3	C13	3	C2	1	See individual bit resets.	64-bit	Activity Monitors Configuration Register
AMCGCR_ELO	3	C13	3	C2	2	See individual bit resets.	64-bit	Activity Monitors Counter Group Configuration Register
AMEVTYPER00_ELO	3	C13	3	C6	0	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_ELO	3	C13	3	C6	1	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_ELO	3	C13	3	C6	2	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_ELO	3	C13	3	C6	3	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0

## 20.5 AMU register summary

The summary table provides an overview of all AMU registers in the core. Individual register descriptions provide detailed information.

**Table 20-3: AMU register summary**

Offset	Name	Reset	Width	Description
0x400	AMEVTYPER00	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x404	AMEVTYPER01	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x408	AMEVTYPER02	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x40C	AMEVTYPER03	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x480	AMEVTYPER10	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x484	AMEVTYPER11	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x488	AMEVTYPER12	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x48C	AMEVTYPER13	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0xCE0	AMCGCR	See individual bit resets.	32-bit	Activity Monitors Counter Group Configuration Register

Offset	Name	Reset	Width	Description
0xE00	<a href="#">AMCFGR</a>	See individual bit resets.	32-bit	Activity Monitors Configuration Register
0xE08	<a href="#">AMIIDR</a>	See individual bit resets.	32-bit	Activity Monitors Implementation Identification Register
0xFBC	<a href="#">AMDEVARCH</a>	See individual bit resets.	32-bit	Activity Monitors Device Architecture Register
0xFCC	<a href="#">AMDEVTYPE</a>	See individual bit resets.	32-bit	Activity Monitors Device Type Register
0xFD0	<a href="#">AMPIDR4</a>	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 4
0xFE0	<a href="#">AMPIDR0</a>	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 0
0xFE4	<a href="#">AMPIDR1</a>	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 1
0xFE8	<a href="#">AMPIDR2</a>	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 2
0xFEC	<a href="#">AMPIDR3</a>	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 3
0xFF0	<a href="#">AMCIDR0</a>	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 0
0xFF4	<a href="#">AMCIDR1</a>	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 1
0xFF8	<a href="#">AMCIDR2</a>	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 2
0xFFC	<a href="#">AMCIDR3</a>	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 3

# Appendix A AArch64 system registers

This appendix contains the descriptions for the Cortex®-X2 AArch64 registers.

## A.1 Generic system control register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Generic system control registers in the core. Individual register descriptions provide detailed information.

**Table A-1: Generic system control register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
AIDR_EL1	3	C0	1	C0	7	0x0	64-bit	Auxiliary ID Register
ACTLR_EL1	3	C1	0	C0	1	0x0	64-bit	Auxiliary Control Register (EL1)
ACTLR_EL2	3	C1	4	C0	1	0x0	64-bit	Auxiliary Control Register (EL2)
HACR_EL2	3	C1	4	C1	7	0x0	64-bit	Hypervisor Auxiliary Control Register
ACTLR_EL3	3	C1	6	C0	1	0x0	64-bit	Auxiliary Control Register (EL3)
AMAIR_EL2	3	C10	0	C3	0	0x0	64-bit	Auxiliary Memory Attribute Indirection Register (EL2)
LORID_EL1	3	C10	0	C4	7	See individual bit resets.	64-bit	LORegionID (EL1)
AMAIR_EL1	3	C10	5	C3	0	0x0	64-bit	Auxiliary Memory Attribute Indirection Register (EL1)
AMAIR_EL3	3	C10	6	C3	0	0x0	64-bit	Auxiliary Memory Attribute Indirection Register (EL3)
RMR_EL3	3	C12	6	C0	2	See individual bit resets.	64-bit	Reset Management Register (EL3)
IMP_CPUACTLR_EL1	3	C15	0	C1	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register (EL1)
IMP_CPUACTLR2_EL1	3	C15	0	C1	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 2 (EL1)
IMP_CPUACTLR3_EL1	3	C15	0	C1	2	See individual bit resets.	64-bit	CPU Auxiliary Control Register 3 (EL1)
IMP_CPUACTLR4_EL1	3	C15	0	C1	3	See individual bit resets.	64-bit	CPU Auxiliary Control Register 4 (EL1)
IMP_CPUECTLR_EL1	3	C15	0	C1	4	See individual bit resets.	64-bit	CPU Extended Control Register
IMP_CPUECTLR2_EL1	3	C15	0	C1	5	See individual bit resets.	64-bit	CPU Extended Control Register 2
IMP_CPUPPMCR3_EL3	3	C15	0	C2	4	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPWRCTLR_EL1	3	C15	0	C2	7	0x0	64-bit	CPU Power Control Register
IMP_ATCR_EL1	3	C15	0	C7	0	0x0	64-bit	CPU Auxiliary Translation Control Register (EL1)
IMP_CPUACTLR5_EL1	3	C15	0	C8	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register 5 (EL1)

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
IMP_CPUACTLR6_EL1	3	C15	0	C8	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 6 (EL1)
IMP_CPUACTLR7_EL1	3	C15	0	C8	2	See individual bit resets.	64-bit	CPU Auxiliary Control Register 7 (EL1)
IMP_ATCR_EL2	3	C15	4	C7	0	0x0	64-bit	CPU Auxiliary Translation Control Register (EL2)
IMP_AVTCR_EL2	3	C15	4	C7	1	0x0	64-bit	CPU Virtualization Auxiliary Translation Control Register (EL2)
IMP_CPUPPMCR_EL3	3	C15	6	C2	0	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR2_EL3	3	C15	6	C2	1	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR4_EL3	3	C15	6	C2	4	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR5_EL3	3	C15	6	C2	5	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUPPMCR6_EL3	3	C15	6	C2	6	See individual bit resets.	64-bit	CPU Power Performance Management Control Register
IMP_CPUACTLR_EL3	3	C15	6	C4	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register (EL3)
IMP_ATCR_EL3	3	C15	6	C7	0	0x0	64-bit	CPU Auxiliary Translation Control Register (EL2)
IMP_CPUPSELR_EL3	3	C15	6	C8	0	See individual bit resets.	64-bit	Selected Instruction Private Select Register
IMP_CPUPCR_EL3	3	C15	6	C8	1	See individual bit resets.	64-bit	Selected Instruction Private Control Register
IMP_CPUPOR_EL3	3	C15	6	C8	2	See individual bit resets.	64-bit	Selected Instruction Private Opcode Register
IMP_CPUPMR_EL3	3	C15	6	C8	3	See individual bit resets.	64-bit	Selected Instruction Private Mask Register
IMP_CPUPOR2_EL3	3	C15	6	C8	4	See individual bit resets.	64-bit	Selected Instruction Private Opcode Register 2
IMP_CPUPMR2_EL3	3	C15	6	C8	5	See individual bit resets.	64-bit	Selected Instruction Private Mask Register 2
IMP_CPUPFR_EL3	3	C15	6	C8	6	See individual bit resets.	64-bit	Selected Instruction Private Flag Register
FPCR	3	C4	3	C4	0	See individual bit resets.	64-bit	Floating-point Control Register
AFSR0_EL2	3	C5	0	C1	0	0x0	64-bit	Auxiliary Fault Status Register 0 (EL2)
AFSR1_EL2	3	C5	0	C1	1	0x0	64-bit	Auxiliary Fault Status Register 1 (EL2)
AFSR0_EL1	3	C5	5	C1	0	0x0	64-bit	Auxiliary Fault Status Register 0 (EL1)
AFSR1_EL1	3	C5	5	C1	1	0x0	64-bit	Auxiliary Fault Status Register 1 (EL1)
AFSR0_EL3	3	C5	6	C1	0	0x0	64-bit	Auxiliary Fault Status Register 0 (EL3)
AFSR1_EL3	3	C5	6	C1	1	0x0	64-bit	Auxiliary Fault Status Register 1 (EL3)

## A.1.1 AIDR\_EL1, Auxiliary ID Register

Provides **IMPLEMENTATION DEFINED** identification information.

The value of this register must be interpreted in conjunction with the value of AArch64-MIDR\_EL1.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

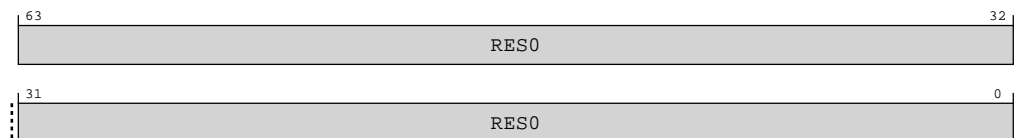
Generic system control

#### Reset value

0x0

### Bit descriptions

**Figure A-1: AArch64\_aidr\_el1 bit assignments**



**Table A-2: AIDR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

### Access

MRS <Xt>, AIDR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AIDR_EL1	0b11	0b001	0b0000	0b0000	0b111

### Accessibility

MRS <Xt>, AIDR\_EL1

```


if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then

```

```
    AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return AIDR_EL1;
    elsif PSTATE.EL == EL2 then
        return AIDR_EL1;
    elsif PSTATE.EL == EL3 then
        return AIDR_EL1;
```

A.1.2 ACTLR\_EL1, Auxiliary Control Register (EL1)

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 AArch64-HCR\_EL2.{E2H, TGE} is {1, 1}, and instead the configuration and control fields are provided by the AArch64-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.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-2: AArch64\_actlr\_el1 bit assignments

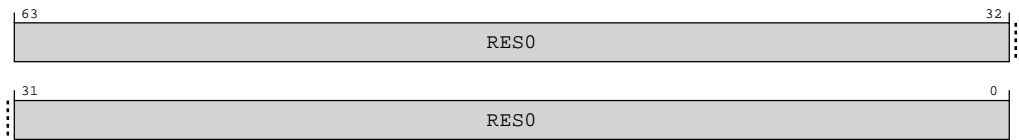


Table A-4: ACTLR\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

## Access

MRS <Xt>, ACTLR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ACTLR_EL1	0b11	0b000	0b0001	0b0000	0b001

MSR ACTLR\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ACTLR_EL1	0b11	0b000	0b0001	0b0000	0b001

## Accessibility

MRS <Xt>, ACTLR\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '1x1' then
        return NVMem[0x118];
    else
        return ACTLR_EL1;
elsif PSTATE.EL == EL2 then
    return ACTLR_EL1;
elsif PSTATE.EL == EL3 then
    return ACTLR_EL1;

```

MSR ACTLR\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TACR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '1x1' then
        NVMem[0x118] = X[t];
    else
        ACTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    ACTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ACTLR_EL1 = X[t];

```

## A.1.3 ACTLR\_EL2, Auxiliary Control Register (EL2)

Provides **IMPLEMENTATION DEFINED** configuration and control options for EL2.



Arm recommends the contents of this register are updated to apply to EL0 when AArch64-HCR\_EL2.{E2H, TGE} is {1, 1}, gaining configuration and control fields from the AArch64-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.



## Configurations

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

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

## Attributes

### Width

64

### Functional group

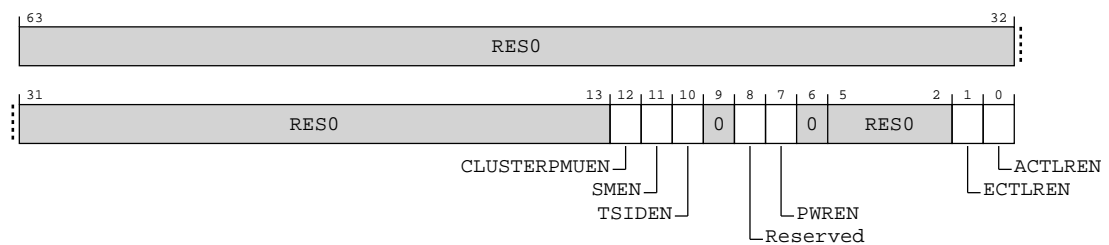
Generic system control

### Reset value

0x0

## Bit descriptions

**Figure A-3: AArch64\_actlr\_el2 bit assignments**



**Table A-7: ACTLR\_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:13]	RES0	Reserved	0x0
[12]	CLUSTERPMUEN	Performance Management Registers enable. The possible values are:  <b>0b0</b> CLUSTERPM* registers are not write-accessible from EL1. This is the reset value.  <b>0b1</b> CLUSTERPM* registers are write-accessible from EL1 if they are write-accessible from EL2.	0x0
[11]	SMEN	Scheme Management Registers enable. The possible values are:  <b>0b0</b> Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible EL1. This is the reset value.  <b>0b1</b> Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible EL1 if they are write-accessible from EL2.	0x0

Bits	Name	Description	Reset
[10]	TSIDEN	Thread Scheme ID Register enable. The possible values are:  <b>0b0</b> Register CLUSTERTHREADSID is not write-accessible from EL1. This is the reset value.  <b>0b1</b> Register CLUSTERTHREADSID is write-accessible from EL1 if they are write-accessible from EL2	0x0
[9]	RES0	Reserved	0x0
[8]	Reserved	Reserved  <b>0b0</b> Reserved  <b>0b1</b> Reserved	0x0
[7]	PWREN	Power Control Registers enable. The possible values are:  <b>0b0</b> Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are not write accessible from EL1. This is the reset value.  <b>0b1</b> Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are write-accessible from EL1 if they are write-accessible from EL2	0x0
[6:2]	RES0	Reserved	0b0000
[1]	ECTLREN	Extended Control Registers enable. The possible values are:  <b>0b0</b> CPUECTLR*_EL1 and CLUSTERECTLR_EL1 are not write-accessible from EL1. This is the reset value.  <b>0b1</b> CPUECTLR*_EL1 and CLUSTERECTLR_EL1 are write-accessible from EL1 if they are write-accessible from EL2.	0b0
[0]	ACTLREN	Auxiliary Control Registers enable. The possible values are:  <b>0b0</b> CPUACTLR*_EL1 and CLUSTERACTLR are not write-accessible from EL1. This is the reset value.  <b>0b1</b> CPUACTLR*_EL1 and CLUSTERACTLR are write-accessible from EL1 if they are write-accessible from EL2	0b0

## Access

MRS <Xt>, ACTLR\_EL2

<systemreg>	op0	op1	CRn	CRm	op2
ACTLR_EL2	0b11	0b100	0b0001	0b0000	0b001

MSR ACTLR\_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ACTLR_EL2	0b11	0b100	0b0001	0b0000	0b001

## Accessibility

MRS <Xt>, ACTLR\_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    return ACTLR_EL2;
elseif PSTATE.EL == EL3 then
    return ACTLR_EL2;

```

MSR ACTLR\_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    ACTLR_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    ACTLR_EL2 = X[t];

```

## A.1.4 HACR\_EL2, Hypervisor Auxiliary Control Register

Controls trapping to EL2 of **IMPLEMENTATION DEFINED** aspects of EL1 or EL0 operation.



Arm recommends that the values in this register do not cause unnecessary traps to EL2 when AArch64-HCR\_EL2.{E2H, TGE} == {1, 1}.

## Configurations

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

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

## Attributes

### Width

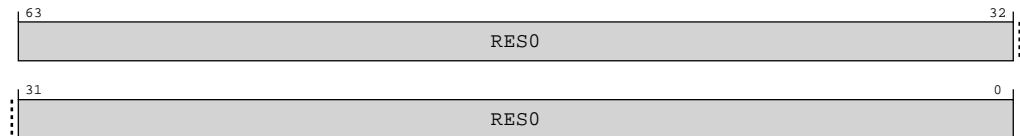
64

**Functional group**

Generic system control

**Reset value**

0x0

**Bit descriptions****Figure A-4: AArch64\_hacr\_el2 bit assignments****Table A-10: HACR\_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

**Access**

MRS &lt;Xt&gt;, HACR\_EL2

<systemreg>	op0	op1	CRn	CRm	op2
HACR_EL2	0b11	0b100	0b0001	0b0001	0b111

MSR HACR\_EL2, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
HACR_EL2	0b11	0b100	0b0001	0b0001	0b111

**Accessibility**

MRS &lt;Xt&gt;, HACR\_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    return HACR_EL2;
elseif PSTATE.EL == EL3 then
    return HACR_EL2;

```

MSR HACR\_EL2, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then

```

```

if EL2Enabled() && HCR_EL2.NV == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    HACR_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    HACR_EL2 = X[t];

```

## A.1.5 ACTLR\_EL3, Auxiliary Control Register (EL3)

Provides **IMPLEMENTATION DEFINED** configuration and control options for EL3.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

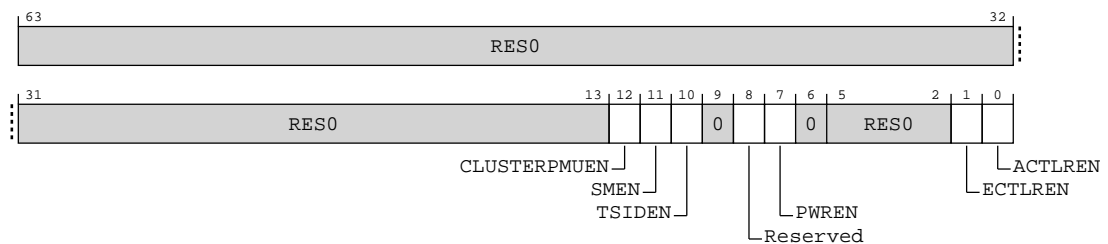
Generic system control

#### Reset value

0x0

### Bit descriptions

**Figure A-5: AArch64\_actlr\_el3 bit assignments**



**Table A-13: ACTLR\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:13]	RES0	Reserved	0x0
[12]	CLUSTERPMUEN	Performance Management Registers enable. The possible values are:  <b>0b0</b> CLUSTERPM* registers are not write-accessible from EL2 and EL1. This is the reset value.  <b>0b1</b> CLUSTERPM* registers are write-accessible from EL2 and EL1 if they are write-accessible from EL2.	0x0

Bits	Name	Description	Reset
[11]	SMEN	<p>Scheme Management Registers enable. The possible values are:</p> <p><b>0b0</b></p> <p>Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are not write-accessible EL2 and EL1. This is the reset value.</p> <p><b>0b1</b></p> <p>Registers CLUSTERACPSID, CLUSTERSTASHSID, CLUSTERPARTCR, CLUSTERBUSQOS, and CLUSTERTHREADSIDOVR are write-accessible EL2 and EL1 if they are write-accessible from EL2.</p>	0x0
[10]	TSIDEN	<p>Thread Scheme ID Register enable. The possible values are:</p> <p><b>0b0</b></p> <p>Register CLUSTERTHREADSID is not write-accessible from EL2 and EL1. This is the reset value.</p> <p><b>0b1</b></p> <p>Register CLUSTERTHREADSID is write-accessible from EL2 and EL1 if they are write-accessible from EL2</p>	0x0
[9]	RESO	Reserved	0x0
[8]	Reserved	<p>Reserved</p> <p><b>0b0</b></p> <p>Reserved</p> <p><b>0b1</b></p> <p>Reserved</p>	0x0
[7]	PWREN	<p>Power Control Registers enable. The possible values are:</p> <p><b>0b0</b></p> <p>Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are not write accessible from EL2 and EL1. This is the reset value.</p> <p><b>0b1</b></p> <p>Registers CPUPWRCTLR, CLUSTERPWRCTLR, CLUSTERPWRDN, CLUSTERPWRSTAT, CLUSTERL3HIT and CLUSTERL3MISS are write-accessible from EL2 and EL1 if they are write-accessible from EL2</p>	0x0
[6:2]	RESO	Reserved	0b0000
[1]	ECTLREN	<p>Extended Control Registers enable. The possible values are:</p> <p><b>0b0</b></p> <p>CPUECTLR*_EL2 and EL1 and CLUSTERECTLR_EL2 and EL1 are not write-accessible from EL2 and EL1. This is the reset value.</p> <p><b>0b1</b></p> <p>CPUECTLR*_EL2 and EL1 and CLUSTERECTLR_EL2 and EL1 are write-accessible from EL2 and EL1 if they are write-accessible from EL2.</p>	0b0
[0]	ACTLREN	<p>Auxiliary Control Registers enable. The possible values are:</p> <p><b>0b0</b></p> <p>CPUACTLR*_EL2 and EL1 and CLUSTERACTLR are not write-accessible from EL2 and EL1. This is the reset value.</p> <p><b>0b1</b></p> <p>CPUACTLR*_EL2 and EL1 and CLUSTERACTLR are write-accessible from EL2 and EL1 if they are write-accessible from EL2</p>	0b0

## Access

MRS <Xt>, ACTLR\_EL3

<systemreg>	op0	op1	CRn	CRm	op2
ACTLR_EL3	0b11	0b110	0b0001	0b0000	0b001

MSR ACTLR\_EL3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ACTLR_EL3	0b11	0b110	0b0001	0b0000	0b001

## Accessibility

MRS <Xt>, ACTLR\_EL3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return ACTLR_EL3;

```

MSR ACTLR\_EL3, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    ACTLR_EL3 = X[t];

```

## A.1.6 AMAIR\_EL2, Auxiliary Memory Attribute Indirection Register (EL2)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by AArch64-MAIR\_EL2.

### Configurations

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

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

### Attributes

#### Width

64

**Functional group**

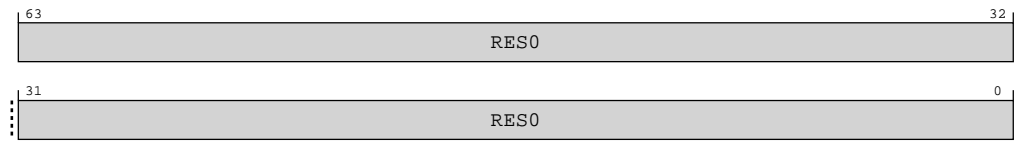
Generic system control

**Reset value**

0x0

**Bit descriptions**

AMAIR\_EL2 is permitted to be cached in a TLB.

**Figure A-6: AArch64\_amair\_el2 bit assignments****Table A-16: AMAIR\_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

**Access**

MRS &lt;Xt&gt;, AMAIR\_EL2

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL2	0b11	0b100	0b1010	0b0011	0b000

MSR AMAIR\_EL2, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL2	0b11	0b100	0b1010	0b0011	0b000

MRS &lt;Xt&gt;, AMAIR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL1	0b11	0b000	0b1010	0b0011	0b000

MSR AMAIR\_EL1, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL1	0b11	0b000	0b1010	0b0011	0b000

**Accessibility**

MRS &lt;Xt&gt;, AMAIR\_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
```



```

elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    return AMAIR_EL2;
elseif PSTATE.EL == EL3 then
    return AMAIR_EL2;

```

## MSR AMAIR\_EL2, &lt;Xt&gt;

```

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

```

## MRS &lt;Xt&gt;, AMAIR\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x148];
    else
        return AMAIR_EL1;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AMAIR_EL2;
    else
        return AMAIR_EL1;
elseif PSTATE.EL == EL3 then
    return AMAIR_EL1;

```

## MSR AMAIR\_EL1, &lt;Xt&gt;

```

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

```

```
elseif PSTATE.EL == EL3 then
    AMAIR_EL1 = X[t];
```

A.1.7 LORID\_EL1, LORegionID (EL1)

Indicates the number of LORegions and LORegion descriptors supported by the PE.

Configurations

If no LORegion descriptors are implemented, then the registers AArch64-LORC\_EL1, AArch64-LORN\_EL1, AArch64-LOREA\_EL1, and AArch64-LORSA\_EL1 are RES0.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets.

Bit descriptions

Figure A-7: AArch64\_lorid\_el1 bit assignments

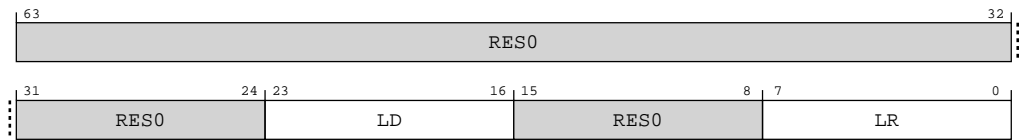


Table A-21: LORID\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	0x0
[23:16]	LD	Number of LORegion descriptors supported by the PE. This is an 8-bit binary number.  <b>0b000000100</b> Four LOR descriptors are supported	
[15:8]	RES0	Reserved	0b00000000
[7:0]	LR	Number of LORegions supported by the PE. This is an 8-bit binary number.  <b>Note:</b> If LORID_EL1 indicates that no LORegions are implemented, then LoadLOAcquire and StoreLORelease will behave as LoadAcquire and StoreRelease.  <b>0b000000100</b> Four LORegions are supported	

## Access

MRS <Xt>, LORID\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
LORID_EL1	0b11	0b000	0b1010	0b0100	0b111

## Accessibility

MRS <Xt>, LORID\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TLOR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.LORID_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TLOR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return LORID_EL1;
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TLOR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return LORID_EL1;
elseif PSTATE.EL == EL3 then
    return LORID_EL1;

```

## A.1.8 AMAIR\_EL1, Auxiliary Memory Attribute Indirection Register (EL1)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by AArch64-MAIR\_EL1.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

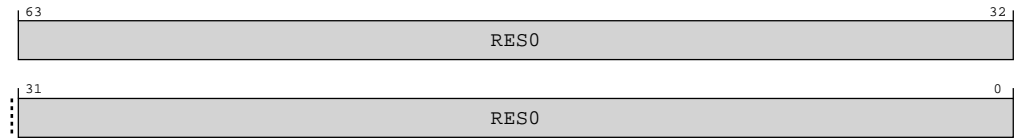
Generic system control

#### Reset value

0x0

### Bit descriptions

AMAIR\_EL1 is permitted to be cached in a TLB.

**Figure A-8: AArch64\_amair\_el1 bit assignments****Table A-23: AMAIR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

**Access**

MRS &lt;Xt&gt;, AMAIR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL1	0b11	0b000	0b1010	0b0011	0b000

MSR AMAIR\_EL1, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL1	0b11	0b000	0b1010	0b0011	0b000

MRS &lt;Xt&gt;, AMAIR\_EL12

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL12	0b11	0b101	0b1010	0b0011	0b000

MSR AMAIR\_EL12, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL12	0b11	0b101	0b1010	0b0011	0b000

**Accessibility**

MRS &lt;Xt&gt;, AMAIR\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AMAIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x148];
    else
        return AMAIR_EL1;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AMAIR_EL2;
    else

```

```

        return AMAIR_EL1;
    elsif PSTATE.EL == EL3 then
        return AMAIR_EL1;

```

## MSR AMAIR\_EL1, &lt;Xt&gt;

```

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

```

## MRS &lt;Xt&gt;, AMAIR\_EL12

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        return NVMem[0x148];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AMAIR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        return AMAIR_EL1;
    else
        UNDEFINED;

```

## MSR AMAIR\_EL12, &lt;Xt&gt;

```

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

```

```
elseif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        AMAIR_EL1 = X[t];
    else
        UNDEFINED;
```

### A.1.9 AMAIR\_EL3, Auxiliary Memory Attribute Indirection Register (EL3)

Provides **IMPLEMENTATION DEFINED** memory attributes for the memory regions specified by AArch64-MAIR\_EL3.

#### Configurations

This register is available in all configurations.

#### Attributes

**Width**

64

**Functional group**

Generic system control

**Reset value**

0x0

#### Bit descriptions

AMAIR\_EL3 is permitted to be cached in a TLB.

Figure A-9: AArch64\_amair\_el3 bit assignments

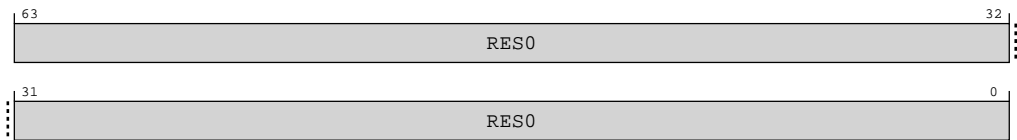


Table A-28: AMAIR\_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

#### Access

MRS <Xt>, AMAIR\_EL3

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL3	0b11	0b110	0b1010	0b0011	0b000

MSR AMAIR\_EL3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL3	0b11	0b110	0b1010	0b0011	0b000

## Accessibility

MRS <Xt>, AMAIR\_EL3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return AMAIR_EL3;

```

MSR AMAIR\_EL3, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    AMAIR_EL3 = X[t];

```

## A.1.10 RMR\_EL3, Reset Management Register (EL3)

A write to the register at EL3 can request a Warm reset.

### Configurations

When EL3 is implemented:

- If EL3 can use all Execution states then this register must be implemented.
- In a AArch64 only implementation it is IMPLEMENTATION DEFINED whether the register is implemented.

Otherwise, direct accesses to RMR\_EL3 are UNDEFINED.

### Attributes

#### Width

64

#### Functional group

Generic system control

#### Reset value

See individual bit resets.

Bit descriptions

Figure A-10: AArch64\_rmr\_el3 bit assignments

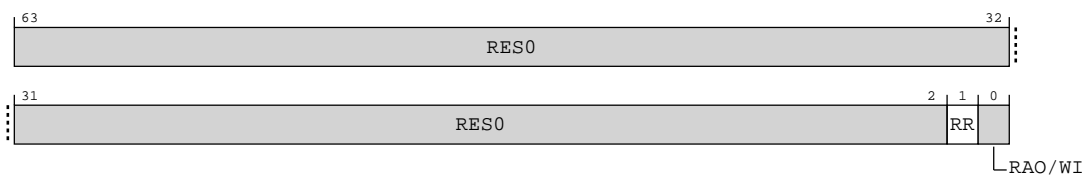


Table A-31: RMR\_EL3 bit descriptions

Bits	Name	Description	Reset
[63:2]	RES0	Reserved	0x0
[1]	RR	Reset Request. Setting this bit to 1 requests a Warm reset.	0x0
[0]	RAO/WI	Reserved	

Access

MRS <Xt>, RMR\_EL3

<systemreg>	op0	op1	CRn	CRm	op2
RMR_EL3	0b11	0b110	0b1100	0b0000	0b010

MSR RMR\_EL3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
RMR_EL3	0b11	0b110	0b1100	0b0000	0b010

Accessibility

MRS <Xt>, RMR\_EL3

```
if PSTATE.EL == EL3 then
    return RMR_EL3;
else
    UNDEFINED;
```

MSR RMR\_EL3, <Xt>

```
if PSTATE.EL == EL3 then
    RMR_EL3 = X[t];
else
    UNDEFINED;
```



### A.1.11 IMP\_CPUACTLR\_EL1, CPU Auxiliary Control Register (EL1)

This register contains control bits that affect the CPU behavior

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

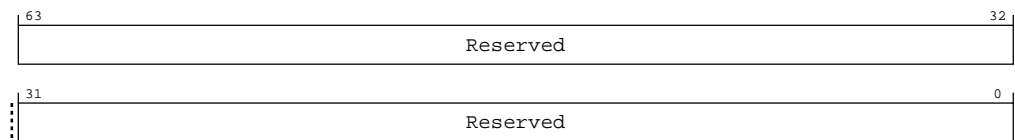
Generic system control

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure A-11: AArch64\_imp\_cpuctlr\_el1 bit assignments**



**Table A-34: IMP\_CPUACTLR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

#### Access

MRS <Xt>, S3\_O\_C15\_C1\_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_O_C15_C1_0	0b11	0b000	0b1111	0b0001	0b000

MSR S3\_O\_C15\_C1\_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_O_C15_C1_0	0b11	0b000	0b1111	0b0001	0b000

#### Accessibility

MRS <Xt>, S3\_O\_C15\_C1\_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then

```

```

    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR_EL1;
    elsif PSTATE.EL == EL2 then
        return IMP_CPUACTLR_EL1;
    elsif PSTATE.EL == EL3 then
        return IMP_CPUACTLR_EL1;

```

MSR S3\_0\_C15\_C1\_0, <Xt>

```

    if PSTATE.EL == EL0 then
        UNDEFINED;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && HCR_EL2.TIDCP == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif ACTLR_EL3.ACTLREN == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            IMP_CPUACTLR_EL1 = X[t];
    elsif PSTATE.EL == EL2 then
        if ACTLR_EL3.ACTLREN == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            IMP_CPUACTLR_EL1 = X[t];
    elsif PSTATE.EL == EL3 then
        IMP_CPUACTLR_EL1 = X[t];

```

## A.1.12 IMP\_CPUACTLR2\_EL1, CPU Auxiliary Control Register 2 (EL1)

This register contains control bits that affect the CPU behavior

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

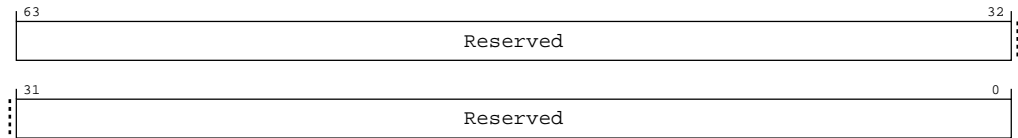
Generic system control

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-12: AArch64\_imp\_cpuctlr2\_el1 bit assignments**



**Table A-37: IMP\_CPUACTLR2\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

### Access

MRS <Xt>, S3\_O\_C15\_C1\_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_O_C15_C1_1	0b11	0b000	0b1111	0b0001	0b001

MSR S3\_O\_C15\_C1\_1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_O_C15_C1_1	0b11	0b000	0b1111	0b0001	0b001

### Accessibility

MRS <Xt>, S3\_O\_C15\_C1\_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR2_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUACTLR2_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR2_EL1;

```

MSR S3\_O\_C15\_C1\_1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ACTLRN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif ACTLR_EL3.ACTLRN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR2_EL1 = X[t];

```

```
elseif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR2_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_CPUACTLR2_EL1 = X[t];
```

### A.1.13 IMP\_CPUACTLR3\_EL1, CPU Auxiliary Control Register 3 (EL1)

This register contains control bits that affect the CPU behavior

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

Generic system control

##### Reset value

See individual bit resets.

#### Bit descriptions

Figure A-13: AArch64\_imp\_cpuactlr3\_el1 bit assignments

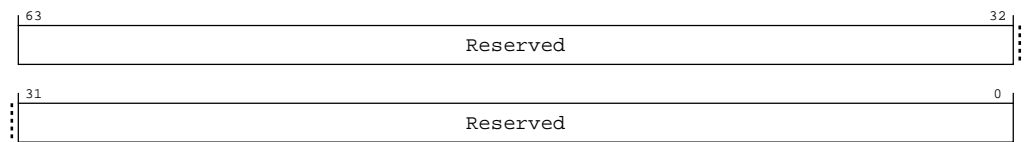


Table A-40: IMP\_CPUACTLR3\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

#### Access

MRS <Xt>, S3\_0\_C15\_C1\_2

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_2	0b11	0b000	0b1111	0b0001	0b010

MSR S3\_0\_C15\_C1\_2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_O_C15_C1_2	0b11	0b000	0b1111	0b0001	0b010

## Accessibility

MRS <Xt>, S3\_O\_C15\_C1\_2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR3_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUACTLR3_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR3_EL1;

```

MSR S3\_O\_C15\_C1\_2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR3_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR3_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_CPUACTLR3_EL1 = X[t];

```

## A.1.14 IMP\_CPUACTLR4\_EL1, CPU Auxiliary Control Register 4 (EL1)

This register contains control bits that affect the CPU behavior

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

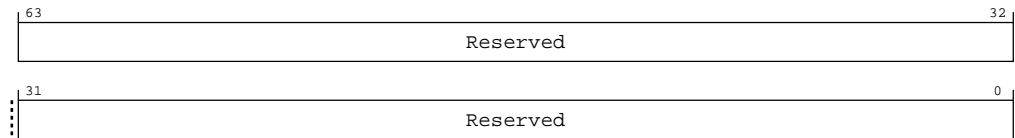
Generic system control

## Reset value

See individual bit resets.

## Bit descriptions

**Figure A-14: AArch64\_imp\_cpuctlr4\_el1 bit assignments**



**Table A-43: IMP\_CPUACTLR4\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

## Access

MRS <Xt>, S3\_0\_C15\_C1\_3

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_3	0b11	0b000	0b1111	0b0001	0b011

MSR S3\_0\_C15\_C1\_3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_3	0b11	0b000	0b1111	0b0001	0b011

## Accessibility

MRS <Xt>, S3\_0\_C15\_C1\_3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR4_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUACTLR4_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUACTLR4_EL1;

```

MSR S3\_0\_C15\_C1\_3, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ACTLRN == '0' then

```

```

        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR4_EL1 = X[t];
    elsif PSTATE_EL == EL2 then
        if ACTLR_EL3.ACTLREN == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            IMP_CPUACTLR4_EL1 = X[t];
    elsif PSTATE_EL == EL3 then
        IMP_CPUACTLR4_EL1 = X[t];

```

## A.1.15 IMP\_CPUACTLR4\_EL1, CPU Extended Control Register

This register contains control bits that affect the CPU behavior

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

Generic system control

#### Reset value

See individual bit resets.

## Bit descriptions

Figure A-15: AArch64\_imp\_cpuctlr\_el1 bit assignments

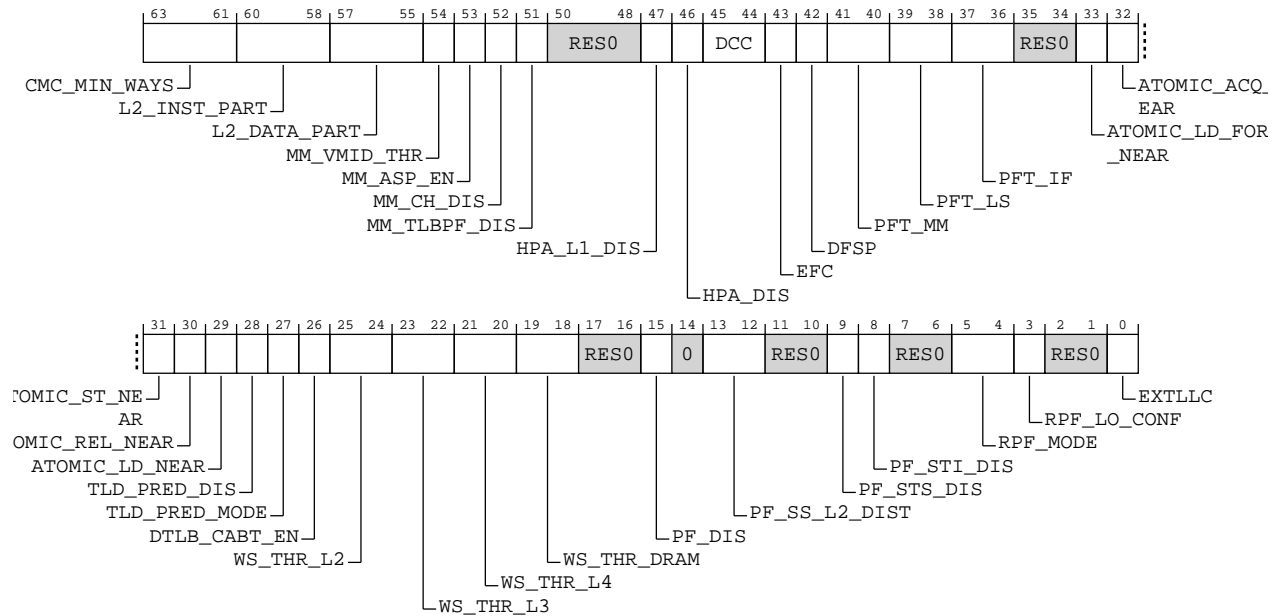


Table A-46: IMP\_CPUCTLR\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:61]	CMC_MIN_WAYS	<p>Limits how many ways of L2 can be used by CMC. The possible values are:</p> <p><b>0b000</b> CMC disabled</p> <p><b>0b001</b> CMC must leave at least 1 way for data in L2</p> <p><b>0b010</b> CMC must leave at least 2 ways for data in L2 - This is the default value.</p> <p><b>0b011</b> CMC must leave at least 3 ways for data in L2</p> <p><b>0b100</b> CMC must leave at least 4 ways for data in L2</p> <p><b>0b101</b> CMC must leave at least 5 ways for data in L2</p> <p><b>0b110</b> CMC must leave at least 6 ways for data in L2</p> <p><b>0b111</b> CMC must leave at least 7 ways for data in L2</p>	



Bits	Name	Description	Reset
[60:58]	L2_INST_PART	<p>Partition the L2 cache for Instruction. The possible values are:</p> <p><b>0b000</b> No ways reserved for instructions. This is the reset value</p> <p><b>0b001</b> Reserve 1 way for instruction. Only instruction fetches can allocate way 7</p> <p><b>0b010</b> Reserve 2 ways for instruction. Only instruction fetches can allocate ways 7:6</p> <p><b>0b011</b> Reserve 3 ways for instruction. Only instruction fetches can allocate ways 7:5</p> <p><b>0b100</b> Reserve 4 ways for instruction. Only instruction fetches can allocate ways 7:4</p> <p><b>0b101</b> Reserve 5 ways for instruction. Only instruction fetches can allocate ways 7:3</p> <p><b>0b110</b> Reserve 6 ways for instruction. Only instruction fetches can allocate ways 7:2</p> <p><b>0b111</b> Reserve 7 ways for instruction. Only instruction fetches can allocate ways 7:1</p>	
[57:55]	L2_DATA_PART	<p>Reserve L2 capacity for data accesses. The possible values are:</p> <p><b>0b000</b> No ways reserved for data. This is the reset value</p> <p><b>0b001</b> Reserve 1 way for data. Only data accesses can allocate way 0</p> <p><b>0b010</b> Reserve 2 ways for data. Only data accesses can allocate ways 1:0</p> <p><b>0b011</b> Reserve 3 ways for data. Only data accesses can allocate ways 2:0</p> <p><b>0b100</b> Reserve 4 ways for data. Only data accesses can allocate ways 3:0</p> <p><b>0b101</b> Reserve 5 ways for data. Only data accesses can allocate ways 4:0</p> <p><b>0b110</b> Reserve 6 ways for data. Only data accesses can allocate ways 5:0</p> <p><b>0b111</b> Reserve 7 ways for data. Only data accesses can allocate ways 6:0</p>	
[54]	MM_VMID_THR	<p>VMID filter threshold. The possible values are:</p> <p><b>0b0</b> VMID filter flush after 16 unique VMID allocations to the MMU Translation Cache. This is the default value.</p> <p><b>0b1</b> VMID filter flush after 32 unique VMID allocations to the MMU Translation Cache</p>	

Bits	Name	Description	Reset
[53]	MM_ASP_EN	Disables allocation of splintered pages in L2 TLB. The possible values are:  <b>0b0</b> Enables allocation of splintered pages in the L2 TLB. This is the default value.  <b>0b1</b> Disables allocation of splintered pages in the L2 TLB.	
[52]	MM_CH_DIS	Disables use of contiguous hint. The possible values are:  <b>0b0</b> Enables use of contiguous hint. This is the default value.  <b>0b1</b> Disables use of contiguous hint.	
[51]	MM_TLBPF_DIS	Disables TLB prefetcher. The possible values are:  <b>0b0</b> Enables TLB prefetcher. This is the default value.  <b>0b1</b> Disables TLB prefetcher.	
[50:48]	RES0	Reserved	0b000
[47]	HPA_L1_DIS	Disables hardware page aggregation in L1 TLBs. The possible values are:  <b>0b0</b> Enables hardware page aggregation in L1 TLBs. This is the default value.  <b>0b1</b> Disables hardware page aggregation in L1 TLBs.	
[46]	HPA_DIS	Disable Hardware page aggregation. The possible values are:  <b>0b0</b> Enables hardware page aggregation. This is the default value.  <b>0b1</b> Disables hardware page aggregation.	
[45:44]	DCC	Downstream Cache Control. Controls whether evictions of clean cache-lines send data on the CHI interface. Set this based on whether there is a cache on the path to memory. The possible values are:  <b>0b00</b> Disables sending data when clean cache-lines are evicted.  <b>0b01</b> Enables sending WriteEvictFull transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data.  <b>0b10</b> Enables sending WriteEvictOrEvict transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data. This is the default value when the SCU is not present  <b>0b11</b> Enables sending WriteEvictOrEvict transactions when Unique Clean or Shared Clean cache-lines are evicted. This is the default value when SCU is present	

Bits	Name	Description	Reset
[43]	EFC	<p>Eviction flush control. Controls whether hardware cache flushes and DC CISC instructions send data when evicting clean cachelines on the CHI interface. The possible values are:</p> <p><b>0b0</b></p> <p>Disables sending data when hardware cache flushes or DC CISC instructions evict a clean cacheline. Sending of Evict transactions is controlled by Downstream Snoop Filter Present (DSFP). This is the default value.</p> <p><b>0b1</b></p> <p>Sending of data when hardware cache flushes or DC CISC instructions evict clean cachelines is controlled by Downstream Cache Control (DCC). Sending of Evict transactions is controlled by Downstream Snoop Filter Present (DSFP).</p>	
[42]	DFSP	<p>Downstream snoop filter present. Enables sending Evict transactions on the CHI when clean cachelines are evicted without data. Enable this if there is at least one snoop filter in the path to memory. The possible values are:</p> <p><b>0b0</b></p> <p>Disables sending Evict transactions when clean cachelines are evicted without data</p> <p><b>0b1</b></p> <p>Enables sending of Evict transactions when clean cachelines are evicted without data. This is the default value</p>	
[41:40]	PFT_MM	<p>DRAM prefetch using PrefetchTgt transactions for tablewalk requests. The possible values are:</p> <p><b>0b00</b></p> <p>Disable prefetchtgt generation for requests from the Memory Management unit (MMU). This is the default value.</p> <p><b>0b01</b></p> <p>Conservatively generate prefetchtgt for cacheable requests from the MMU, always generate for Non-cacheable.</p> <p><b>0b10</b></p> <p>Aggressively generate prefetchtgt for cacheable requests from the MMU, always generate for Non-cacheable.</p> <p><b>0b11</b></p> <p>Always generate prefetchtgt for cacheable requests from the MMU, always generate for Non-cacheable.</p>	
[39:38]	PFT_LS	<p>DRAM prefetch using PrefetchTgt transactions for load and store requests. The possible values are:</p> <p><b>0b00</b></p> <p>Disable prefetchtgt generation for requests from the Load-Store unit (LS). This is the default value.</p> <p><b>0b01</b></p> <p>Conservatively generate prefetchtgt for cacheable requests from the LS, always generate for Non-cacheable.</p> <p><b>0b10</b></p> <p>Aggressively generate prefetchtgt for cacheable requests from the LS, always generate for Non-cacheable.</p> <p><b>0b11</b></p> <p>Always generate prefetchtgt for cacheable requests from the LS, always generate for Non-cacheable.</p>	

Bits	Name	Description	Reset
[37:36]	PFT_IF	<p>DRAM prefetch using PrefetchTgt transactions for instruction fetch requests. The possible values are:</p> <p><b>0b00</b></p> <p>Disable prefetchtgt generation for requests from the Instruction Fetch unit (IF). This is the default value.</p> <p><b>0b01</b></p> <p>Conservatively generate prefetchtgt for cacheable requests from the IF, always generate for Non-cacheable.</p> <p><b>0b10</b></p> <p>Aggressively generate prefetchtgt for cacheable requests from the IF, always generate for Non-cacheable.</p> <p><b>0b11</b></p> <p>Always generate prefetchtgt for cacheable requests from the IF, always generate for Non-cacheable.</p>	
[35:34]	RES0	Reserved	0b00
[33]	ATOMIC_LD_FORCE_NEAR	<p>A load atomic (including SWP &amp; CAS) instruction to WB memory will be performed near. The possible values are:</p> <p><b>0b0</b></p> <p>Load-atomic is near if cache line is already Exclusive, otherwise make far atomic request.</p> <p><b>0b1</b></p> <p>Load-atomic will be performed near by bringing the line into the L1D Cache. This is the default value.</p>	
[32]	ATOMIC_ACQ_NEAR	<p>An atomic instruction to WB memory with acquire semantics that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are:</p> <p><b>0b0</b></p> <p>Acquire-atomic is near if cache line is already Exclusive, otherwise make far atomic request.</p> <p><b>0b1</b></p> <p>Acquire-atomic will make up to 1 fill request to perform near. This is the default value.</p>	
[31]	ATOMIC_ST_NEAR	<p>A store atomic instruction to WB memory that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are:</p> <p><b>0b0</b></p> <p>Store-atomic is near if cache line is already Exclusive, otherwise make far atomic request. This is the default value.</p> <p><b>0b1</b></p> <p>Store-atomic will make up to 1 fill request to perform near.</p>	
[30]	ATOMIC_REL_NEAR	<p>An atomic instruction to WB memory with release semantics that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are:</p> <p><b>0b0</b></p> <p>Release-atomic is near if cache line is already Exclusive, otherwise make far atomic request.</p> <p><b>0b1</b></p> <p>Release-atomic will make up to 1 fill request to perform near. This is the default value.</p>	

Bits	Name	Description	Reset
[29]	ATOMIC_LD_NEAR	<p>A load atomic (including SWP &amp; CAS) instruction to WB memory that does not hit in the cache in Exclusive state, may make up to one fill request. The possible values are:</p> <p><b>0b0</b> Load-atomic is near if cache line is already Exclusive, otherwise make far atomic request. This is the default value.</p> <p><b>0b1</b> Load-atomic will make up to 1 fill request to perform near.</p>	
[28]	TLD_PRED_DIS	<p>Disable Transient Load Prediction. The possible values are:</p> <p><b>0b0</b> Enables transient load prediction. This is the default value.</p> <p><b>0b1</b> Disables transient load prediction.</p>	
[27]	TLD_PRED_MODE	<p>Aggressive Transient Load Prediction. The possible values are:</p> <p><b>0b0</b> Disables aggressive transient load prediction. This is the default value.</p> <p><b>0b1</b> Enables aggressive transient load prediction.</p>	
[26]	DTLB_CABT_EN	<p>Enables TLB Conflict Data Abort Exception. The possible values are:</p> <p><b>0b0</b> Disables TLB conflict data abort exception. This is the default value.</p> <p><b>0b1</b> Enables TLB conflict data abort exception.</p>	
[25:24]	WS_THR_L2	<p>Threshold for direct stream to L2 cache on store. The possible values are:</p> <p><b>0b00</b> 256B - This is the default value</p> <p><b>0b01</b> 4KB</p> <p><b>0b10</b> 8KB</p> <p><b>0b11</b> Disables direct stream to L2 cache on store.</p>	
[23:22]	WS_THR_L3	<p>Threshold for direct stream to L3 cache on store. The possible values are:</p> <p><b>0b00</b> 128KB</p> <p><b>0b01</b> 256KB - This is the default value</p> <p><b>0b10</b> 512KB</p> <p><b>0b11</b> Disables direct stream to L3 cache on store.</p>	

Bits	Name	Description	Reset
[21:20]	WS_THR_L4	Threshold for direct stream to L4 cache on store. The possible values are:  <b>0b00</b> 256KB  <b>0b01</b> 512KB - This is the default value  <b>0b10</b> 1MB  <b>0b11</b> Disables direct stream to L4 cache on store.	
[19:18]	WS_THR_DRAM	Threshold for direct stream to DRAM on store. The possible values are:  <b>0b00</b> 512KB  <b>0b01</b> 1MB - This is the default value  <b>0b10</b> 2MB  <b>0b11</b> Disables direct stream to DRAM on store.	
[17:16]	RES0	Reserved	0b00
[15]	PF_DIS	Disables hardware prefetching. The possible values are:  <b>0b0</b> Enables hardware prefetching. This is the default value.  <b>0b1</b> Disables hardware prefetching.	
[14]	RES0	Reserved	0b0
[13:12]	PF_SS_L2_DIST	Single cache line stride prefetching L2 distance. The possible values are:  <b>0b00</b> 22 lines ahead  <b>0b01</b> 40 lines ahead  <b>0b10</b> 60 lines ahead  <b>0b11</b> Dynamic. This is the default value.	
[11:10]	RES0	Reserved	0b00
[9]	PF_STS_DIS	Disable store-stride prefetches. The possible values are:  <b>0b0</b> Enables store prefetching. This is the default value.  <b>0b1</b> Disables store prefetching.	

Bits	Name	Description	Reset
[8]	PF_STI_DIS	Disable store prefetches at issue (not overridden by <code>Is_hw_pref_disable</code> ). The possible values are:  <b>0b0</b> Enables store prefetching. This is the default value.  <b>0b1</b> Disable store prefetching.	
[7:6]	RES0	Reserved	0b00
[5:4]	RPF_MODE	Region prefetcher aggressiveness. The possible values are:  <b>0b00</b> Dynamic region prefetch aggressiveness. This is the default value.  <b>0b01</b> Conservative region prefetching.  <b>0b10</b> Very Conservative region prefetching.  <b>0b11</b> Most Conservative region prefetching. This will disable the region prefetcher.	
[3]	RPF_LO_CONF	Region Prefetcher single accesses training behavior. The possible values are:  <b>0b0</b> Mostly don't train PHT on single access. This is the default value.  <b>0b1</b> Always train the PHT on single access. This results in fewer prefetch requests.	
[2:1]	RES0	Reserved	0b00
[0]	EXTLLC	Internal or external Last-level cache (LLC) in the system. The possible values are:  <b>0b0</b> Indicates that an internal Last-level cache is present in the system, and that the <code>DataSource</code> field on the master CHI interface indicates when data is returned from the LLC. This is used to control how the <code>LL_CACHE*</code> PMU events count. This is the default value.  <b>0b1</b> Indicates that an external Last-level cache is present in the system, and that the <code>DataSource</code> field on the master CHI interface indicates when data is returned from the LLC. This is used to control how the <code>LL_CACHE*</code> PMU events count.	

## Access

MRS <Xt>, S3\_0\_C15\_C1\_4

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_4	0b11	0b000	0b1111	0b0001	0b100

MSR S3\_0\_C15\_C1\_4, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_4	0b11	0b000	0b1111	0b0001	0b100

## Accessibility

MRS <Xt>, S3\_0\_C15\_C1\_4

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUECTLR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUECTLR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUECTLR_EL1;

```

MSR S3\_0\_C15\_C1\_4, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ECTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUECTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUECTLR_EL1 = X[t];

```

### A.1.16 IMP\_CPUECTLR2\_EL1, CPU Extended Control Register 2

This register contains control bits that affect the CPU behavior

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

Generic system control

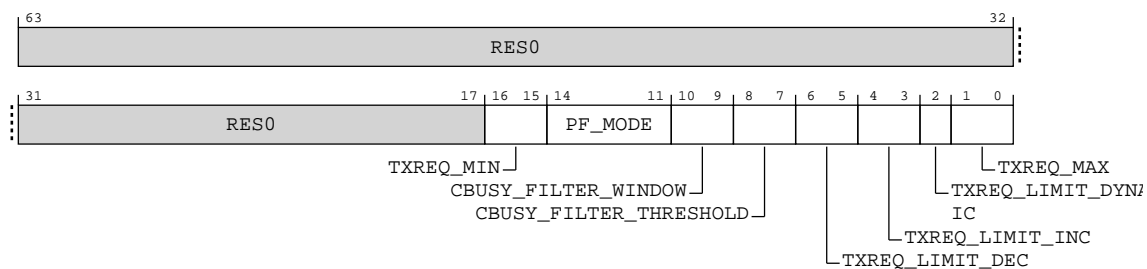
##### Reset value

See individual bit resets.



## Bit descriptions

**Figure A-16: AArch64\_imp\_cpuctlr2\_el1 bit assignments**



### Table A-49: IMP\_CPUCTRL2\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:17]	RES0	Reserved	0x0
[16:15]	TXREQ_MIN	<p>Minimum number of TXREQ transactions outstanding from the L2 Transaction Queue. The possible values are:</p> <p><b>0b00</b> 1/4 of L2 TQ size - This is the default value</p> <p><b>0b01</b> 1/8 of L2 TQ size</p> <p><b>0b10</b> 1/16 of L2 TQ size</p> <p><b>0b11</b> 1/32 of L2 TQ size</p>	

Bits	Name	Description	Reset
[14:11]	PF_MODE	<p>Prefetcher Aggressiveness Modes. With mode 0 representing the most aggressive mode and 3 representing the most conservative mode. The possible values and associated ranges are:</p> <p><b>0b0000</b> Modes [0,0] (statically at the most aggressive mode)</p> <p><b>0b0001</b> Modes [0,1]</p> <p><b>0b0010</b> Modes [0,2]</p> <p><b>0b0011</b> Modes [0,3] - This is the default value.</p> <p><b>0b0100</b> Modes [1,1]</p> <p><b>0b0101</b> Modes [1,2]</p> <p><b>0b0110</b> Modes [1,3]</p> <p><b>0b0111</b> Modes [2,2]</p> <p><b>0b1000</b> Modes [2,3]</p> <p><b>0b1001</b> Modes [3,3] (statically at the most conservative mode)</p> <p><b>0b1010</b> reserved</p> <p><b>0b1011</b> reserved</p> <p><b>0b1100</b> reserved</p> <p><b>0b1101</b> reserved</p> <p><b>0b1110</b> reserved</p> <p><b>0b1111</b> reserved</p>	

Bits	Name	Description	Reset
[10:9]	CBUSY_FILTER_WINDOW	<p>Number of CBusy responses in one sampling window. The possible values are:</p> <p><b>0b00</b> 256 - This is the default value</p> <p><b>0b01</b> 64</p> <p><b>0b10</b> 128</p> <p><b>0b11</b> 512</p>	
[8:7]	CBUSY_FILTER_THRESHOLD	<p>Fraction of of CBusy responses in the sampling window necessary to be considered a valid sample of that CBusy value. The possible values are:</p> <p><b>0b00</b> 1/16 - This is the default value</p> <p><b>0b01</b> 1/32</p> <p><b>0b10</b> 1/8</p> <p><b>0b11</b> 1/4</p>	
[6:5]	TXREQ_LIMIT_DEC	<p>Dynamic TXREQ limit decrement. Controls how quickly the dynamic TXREQ limit is decreased when CBusy indicates value of 3. The possible values are:</p> <p><b>0b00</b> 4 - This is the default value</p> <p><b>0b01</b> 8</p> <p><b>0b10</b> 16</p> <p><b>0b11</b> 2</p>	
[4:3]	TXREQ_LIMIT_INC	<p>Dynamic TXREQ limit increment. Controls how quickly the dynamic TXREQ limit is increased when CBusy indicates values less than 2. The possible values are:</p> <p><b>0b00</b> 4 - This is the default value</p> <p><b>0b01</b> 8</p> <p><b>0b10</b> 16</p> <p><b>0b11</b> 2</p>	

Bits	Name	Description	Reset
[2]	TXREQ_LIMIT_DYNAMIC	<p>Selects static or dynamic control of TXREQ limit. Dynamic TXREQ limit will adjust based on CBusy responses on RXDAT and RXRSP in the range of the static limit selected by CPUECTLR2_EL1[1:0] and 1/4 of the L2 TQ SIZE. The possible values are:</p> <p><b>0b0</b> maximum number of TXREQ transactions statically set by CPUECTLR2_EL1[1:0] - This is the default value.</p> <p><b>0b1</b> maximum number of TXREQ transactions dynamically controlled</p>	
[1:0]	TXREQ_MAX	<p>Maximum number of TXREQ transactions outstanding from the L2 Transaction Queue. The possible values are:</p> <p><b>0b00</b> full L2 TQ size - This is the default value</p> <p><b>0b01</b> 3/4 of L2 TQ size</p> <p><b>0b10</b> 1/2 of L2 TQ size</p> <p><b>0b11</b> 1/4 of L2 TQ size</p>	

### Access

MRS <Xt>, S3\_0\_C15\_C1\_5

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_5	0b11	0b000	0b1111	0b0001	0b101

MSR S3\_0\_C15\_C1\_5, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_5	0b11	0b000	0b1111	0b0001	0b101

### Accessibility

MRS <Xt>, S3\_0\_C15\_C1\_5

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUECTLR2_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUECTLR2_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUECTLR2_EL1;

```

MSR S3\_0\_C15\_C1\_5, <Xt>

```

if PSTATE.EL == EL0 then

```

```
    UNDEFINED;
  elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ECTLREN == '0' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    elseif ACTLR_EL3.ECTLREN == '0' then
      AArch64.SystemAccessTrap(EL3, 0x18);
    else
      IMP_CPUECTLR2_EL1 = X[t];
  elseif PSTATE.EL == EL2 then
    if ACTLR_EL3.ECTLREN == '0' then
      AArch64.SystemAccessTrap(EL3, 0x18);
    else
      IMP_CPUECTLR2_EL1 = X[t];
  elseif PSTATE.EL == EL3 then
    IMP_CPUECTLR2_EL1 = X[t];
```

### A.1.17 IMP\_CPUPPMCR3\_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

Generic system control

##### Reset value

See individual bit resets.

#### Bit descriptions

Figure A-17: AArch64\_imp\_cpuppmcr3\_el3 bit assignments

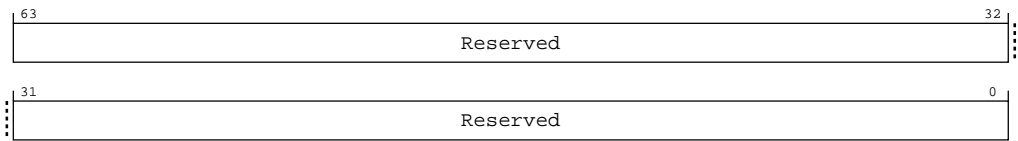


Table A-52: IMP\_CPUPPMCR3\_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

## Access

MRS <Xt>, S3\_0\_C15\_C2\_4

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_4	0b11	0b000	0b1111	0b0010	0b100

MSR S3\_0\_C15\_C2\_4, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_4	0b11	0b000	0b1111	0b0010	0b100

## Accessibility

MRS <Xt>, S3\_0\_C15\_C2\_4

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR3_EL3;

```

MSR S3\_0\_C15\_C2\_4, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPPMCR3_EL3 = X[t];

```

## A.1.18 IMP\_CPUPWRCTLR\_EL1, CPU Power Control Register

This register controls various power aspects of the core.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

Generic system control

#### Reset value

0x0

Bit descriptions

Figure A-18: AArch64\_imp\_cpupwrctrl\_el1 bit assignments

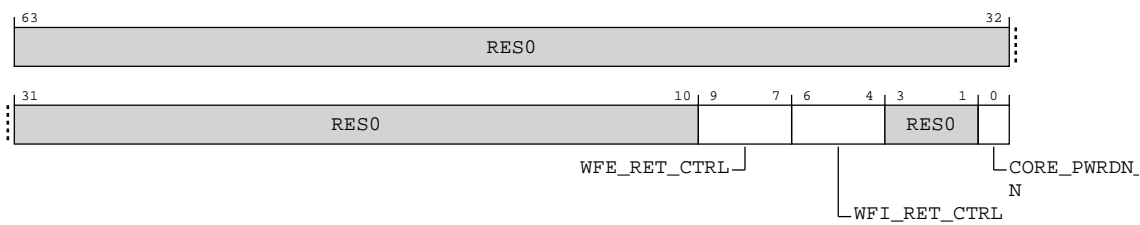


Table A-55: IMP\_CPUPWRCTLR\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:10]	RES0	Reserved	0x0
[9:7]	WFE_RET_CTRL	Wait for Event retention control. The possible values are:  <b>0b000</b> Dynamic retention is disabled.  <b>0b001</b> 2 system counter ticks are required before retention entry.  <b>0b010</b> 8 system counter ticks are required before retention entry.  <b>0b011</b> 32 system counter ticks are required before retention entry.  <b>0b100</b> 64 system counter ticks are required before retention entry.  <b>0b101</b> 128 system counter ticks are required before retention entry.  <b>0b110</b> 256 system counter ticks are required before retention entry.  <b>0b111</b> 512 system counter ticks are required before retention entry.	0x0

Bits	Name	Description	Reset
[6:4]	WFI_RET_CTRL	Wait for Interrupt retention control. The possible values are:  <b>0b000</b> Dynamic retention is disabled.  <b>0b001</b> 2 system counter ticks are required before retention entry.  <b>0b010</b> 8 system counter ticks are required before retention entry.  <b>0b011</b> 32 system counter ticks are required before retention entry.  <b>0b100</b> 64 system counter ticks are required before retention entry.  <b>0b101</b> 128 system counter ticks are required before retention entry.  <b>0b110</b> 256 system counter ticks are required before retention entry.  <b>0b111</b> 512 system counter ticks are required before retention entry.	0x0
[3:1]	RESO	Reserved	0b000
[0]	CORE_PWRDN_EN	Indicates to the power controller if the CPU wants to power down when it enters WFE/WFI state. The possible values are:  <b>0b0</b> CPU does not want to power down when it enters WFE/WFI state.  <b>0b1</b> CPU wants to power down when it enters WFE/WFI state.	0b0

## Access

MRS <Xt>, S3\_0\_C15\_C2\_7

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_7	0b11	0b000	0b1111	0b0010	0b111

MSR S3\_0\_C15\_C2\_7, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C2_7	0b11	0b000	0b1111	0b0010	0b111

## Accessibility

MRS <Xt>, S3\_0\_C15\_C2\_7

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else

```



```

        return IMP_CPUPWRCTLR_EL1;
    elsif PSTATE.EL == EL2 then
        return IMP_CPUPWRCTLR_EL1;
    elsif PSTATE.EL == EL3 then
        return IMP_CPUPWRCTLR_EL1;

```

MSR S3\_0\_C15\_C2\_7, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && ACTLR_EL2.PWREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.PWREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUPWRCTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.PWREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUPWRCTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUPWRCTLR_EL1 = X[t];

```

### A.1.19 IMP\_ATCR\_EL1, CPU Auxiliary Translation Control Register (EL1)

This register contains control bits that affect the CPU behavior.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

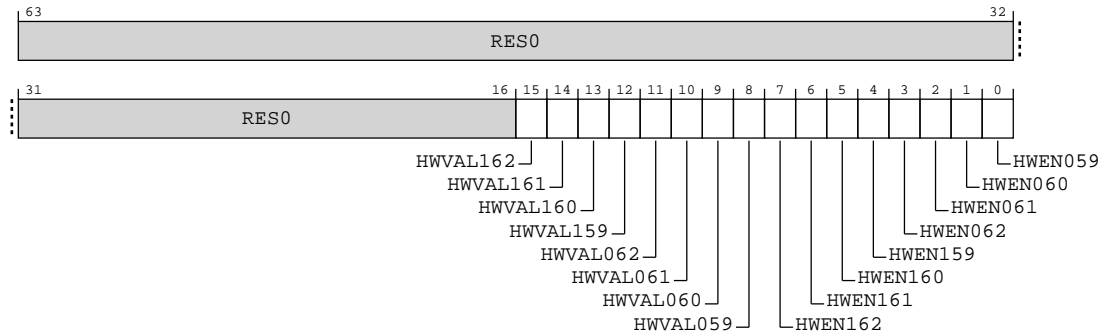
Generic system control

##### Reset value

0x0

## Bit descriptions

**Figure A-19: AArch64\_imp\_atcr\_el1 bit assignments**



**Table A-58: IMP\_ATCR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to translation table walks using TTBR1_EL1 if HWEN162 is set.	0x0
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to translation table walks using TTBR1_EL1 if HWEN161 is set.	0x0
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to translation table walks using TTBR1_EL1 if HWEN160 is set.	0x0
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to translation table walks using TTBR1_EL1 if HWEN159 is set.	0x0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to translation table walks using TTBRO_EL1 if HWEN062 is set.	0x0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to translation table walks using TTBRO_EL1 if HWEN061 is set.	0x0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to translation table walks using TTBRO_EL1 if HWEN060 is set.	0x0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to translation table walks using TTBRO_EL1 if HWEN059 is set.	0x0
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR1_EL1. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0x0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR1_EL1. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0x0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR1_EL1. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0x0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR1_EL1. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0x0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBRO_EL1. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0x0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBRO_EL1. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0x0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBRO_EL1. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0x0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBRO_EL1. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0x0

## Access

MRS <Xt>, S3\_0\_C15\_C7\_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_O_C15_C7_0	0b11	0b000	0b1111	0b0111	0b000

MSR S3\_O\_C15\_C7\_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_O_C15_C7_0	0b11	0b000	0b1111	0b0111	0b000

## Accessibility

MRS <Xt>, S3\_O\_C15\_C7\_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_ATCR_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_ATCR_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_ATCR_EL1;

```

MSR S3\_O\_C15\_C7\_0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_ATCR_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    IMP_ATCR_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_ATCR_EL1 = X[t];

```

## A.1.20 IMP\_CPUACTLR5\_EL1, CPU Auxiliary Control Register 5 (EL1)

This register contains control bits that affect the CPU behavior

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

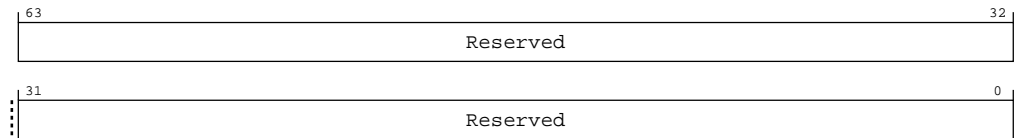
Generic system control

## Reset value

See individual bit resets.

## Bit descriptions

**Figure A-20: AArch64\_imp\_cpuctlr5\_el1 bit assignments**



**Table A-61: IMP\_CPUACTLR5\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

## Access

MRS <Xt>, S3\_0\_C15\_C8\_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C8_0	0b11	0b000	0b1111	0b1000	0b000

MSR S3\_0\_C15\_C8\_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C8_0	0b11	0b000	0b1111	0b1000	0b000

## Accessibility

MRS <Xt>, S3\_0\_C15\_C8\_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR5_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUACTLR5_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR5_EL1;

```

MSR S3\_0\_C15\_C8\_0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ACTLRN == '0' then

```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR5_EL1 = X[t];
    elsif PSTATE_EL == EL2 then
        if ACTLR_EL3.ACTLREN == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            IMP_CPUACTLR5_EL1 = X[t];
    elsif PSTATE_EL == EL3 then
        IMP_CPUACTLR5_EL1 = X[t];
```

A.1.21 IMP\_CPUACTLR6\_EL1, CPU Auxiliary Control Register 6 (EL1)

This register contains control bits that affect the CPU behavior

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets.

Bit descriptions

Figure A-21: AArch64\_imp\_cpuactlr6\_el1 bit assignments

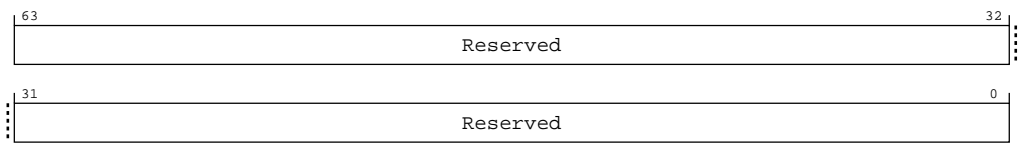


Table A-64: IMP\_CPUACTLR6\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3\_0\_C15\_C8\_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C8_1	0b11	0b000	0b1111	0b1000	0b001

MSR S3\_0\_C15\_C8\_1, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C8_1	0b11	0b000	0b1111	0b1000	0b001

### Accessibility

MRS &lt;Xt&gt;, S3\_0\_C15\_C8\_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR6_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUACTLR6_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR6_EL1;

```

MSR S3\_0\_C15\_C8\_1, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR6_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR6_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    IMP_CPUACTLR6_EL1 = X[t];

```

## A.1.22 IMP\_CPUACTLR7\_EL1, CPU Auxiliary Control Register 7 (EL1)

This register contains control bits that affect the CPU behavior

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

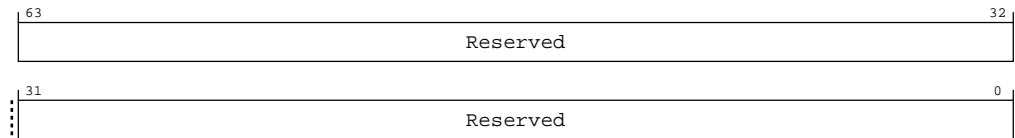
Generic system control

## Reset value

See individual bit resets.

## Bit descriptions

**Figure A-22: AArch64\_imp\_cpuctlr7\_el1 bit assignments**



**Table A-67: IMP\_CPUACTLR7\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

## Access

MRS <Xt>, S3\_0\_C15\_C8\_2

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C8_2	0b11	0b000	0b1111	0b1000	0b010

MSR S3\_0\_C15\_C8\_2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C8_2	0b11	0b000	0b1111	0b1000	0b010

## Accessibility

MRS <Xt>, S3\_0\_C15\_C8\_2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUACTLR7_EL1;
elseif PSTATE.EL == EL2 then
    return IMP_CPUACTLR7_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR7_EL1;

```

MSR S3\_0\_C15\_C8\_2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && ACTLR_EL2.ACTLRN == '0' then

```

```

    AArch64.SystemAccessTrap(EL2, 0x18);
elseif ACTLR_EL3.ACTLREN == '0' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    IMP_CPUACTLR7_EL1 = X[t];
elseif PSTATE_EL == EL2 then
    if ACTLR_EL3.ACTLREN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR7_EL1 = X[t];
elseif PSTATE_EL == EL3 then
    IMP_CPUACTLR7_EL1 = X[t];

```

### A.1.23 IMP\_ATCR\_EL2, CPU Auxiliary Translation Control Register (EL2)

This register contains control bits that affect the CPU behavior.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

Generic system control

##### Reset value

0x0

#### Bit descriptions

Figure A-23: AArch64\_imp\_atcr\_el2 bit assignments

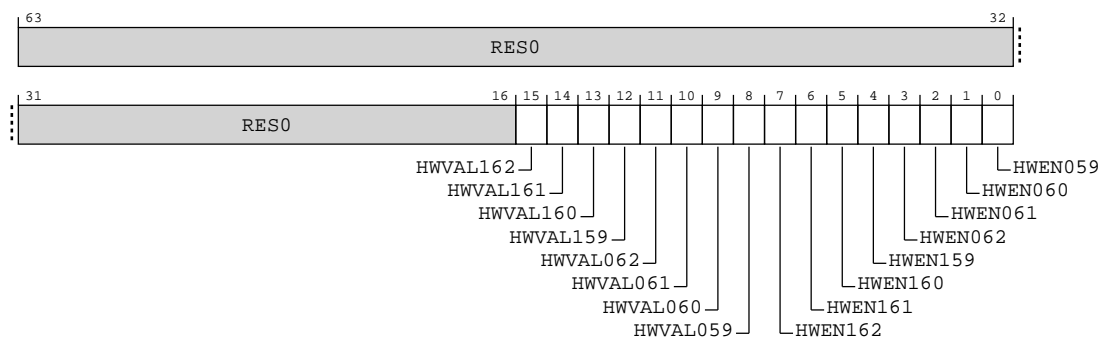


Table A-70: IMP\_ATCR\_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to translation table walks using TTBR1_EL2 if HWEN162 is set.	0x0
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to translation table walks using TTBR1_EL2 if HWEN161 is set.	0x0



Bits	Name	Description	Reset
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to translation table walks using TTBR1_EL2 if HWEN160 is set.	0x0
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to translation table walks using TTBR1_EL2 if HWEN159 is set.	0x0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN062 is set.	0x0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN061 is set.	0x0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN060 is set.	0x0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2 if HWEN059 is set.	0x0
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR1_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0x0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR1_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0x0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR1_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0x0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR1_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0x0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0x0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0x0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0x0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0x0

## Access

MRS <Xt>, S3\_4\_C15\_C7\_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_4_C15_C7_0	0b11	0b100	0b1111	0b0111	0b000

MSR S3\_4\_C15\_C7\_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_4_C15_C7_0	0b11	0b100	0b1111	0b0111	0b000

## Accessibility

MRS <Xt>, S3\_4\_C15\_C7\_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    return IMP_ATCR_EL2;
elseif PSTATE.EL == EL3 then
    return IMP_ATCR_EL2;

```

MSR S3\_4\_C15\_C7\_0, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_ATCR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL2 = X[t];

```

## A.1.24 IMP\_AVTCR\_EL2, CPU Virtualization Auxiliary Translation Control Register (EL2)

This register contains control bits that affect the CPU behavior.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

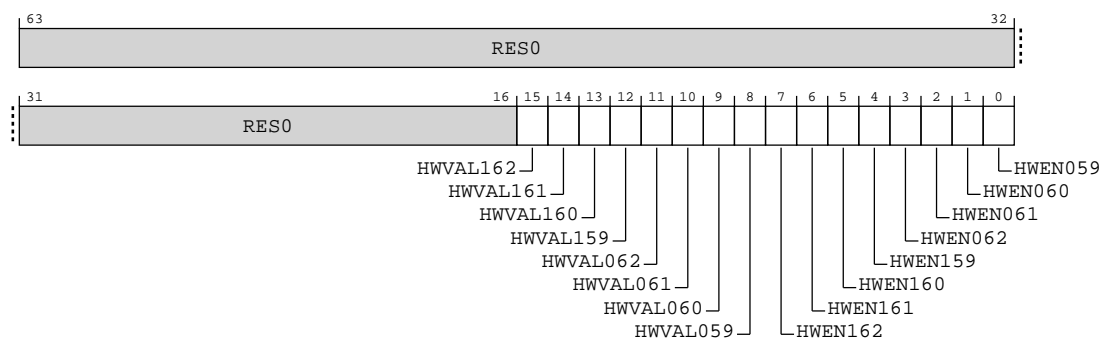
Generic system control

#### Reset value

0x0

### Bit descriptions

Figure A-24: AArch64\_imp\_avtcr\_el2 bit assignments



**Table A-73: IMP\_AVTCR\_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN162 is set.	0x0
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN161 is set.	0x0
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN160 is set.	0x0
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to translation table walks using VSTTBR_EL2 if HWEN159 is set.	0x0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to translation table walks using TTBRO_EL2 if HWEN062 is set.	0x0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to translation table walks using TTBRO_EL2 if HWEN061 is set.	0x0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to translation table walks using TTBRO_EL2 if HWEN060 is set.	0x0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to translation table walks using TTBRO_EL2 if HWEN059 is set.	0x0
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0x0
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0x0
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0x0
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to translation table walks using VSTTBR_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0x0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBRO_EL2. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0x0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBRO_EL2. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0x0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBRO_EL2. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0x0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBRO_EL2. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0x0

**Access**

MRS &lt;Xt&gt;, S3\_4\_C15\_C7\_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_4_C15_C7_1	0b11	0b100	0b1111	0b0111	0b001

MSR S3\_4\_C15\_C7\_1, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
S3_4_C15_C7_1	0b11	0b100	0b1111	0b0111	0b001

## Accessibility

MRS <Xt>, S3\_4\_C15\_C7\_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return IMP_AVTCR_EL2;
elsif PSTATE.EL == EL3 then
    return IMP_AVTCR_EL2;

```

MSR S3\_4\_C15\_C7\_1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    IMP_AVTCR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_AVTCR_EL2 = X[t];

```

## A.1.25 IMP\_CPUPPMCR\_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

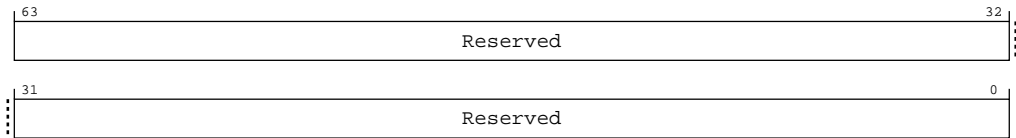
Generic system control

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-25: AArch64\_imp\_cpuppmcr\_el3 bit assignments**



**Table A-76: IMP\_CPUPPMCR\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

### Access

MRS <Xt>, S3\_6\_C15\_C2\_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_0	0b11	0b110	0b1111	0b0010	0b000

MSR S3\_6\_C15\_C2\_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_0	0b11	0b110	0b1111	0b0010	0b000

### Accessibility

MRS <Xt>, S3\_6\_C15\_C2\_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR_EL3;

```

MSR S3\_6\_C15\_C2\_0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR_EL3 = X[t];

```

## A.1.26 IMP\_CPUPPMCR2\_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

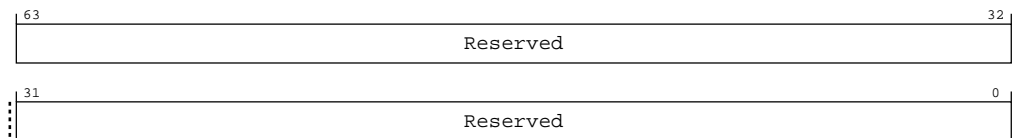
Generic system control

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-26: AArch64\_imp\_cpuppmcr2\_el3 bit assignments**



**Table A-79: IMP\_CPUPPMCR2\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

### Access

MRS <Xt>, S3\_6\_C15\_C2\_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_1	0b11	0b110	0b1111	0b0010	0b001

MSR S3\_6\_C15\_C2\_1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_1	0b11	0b110	0b1111	0b0010	0b001

### Accessibility

MRS <Xt>, S3\_6\_C15\_C2\_1

```
if PSTATE.EL == EL0 then
```

```
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR2_EL3;
```

MSR S3\_6\_C15\_C2\_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPPMCR2_EL3 = X[t];
```

### A.1.27 IMP\_CPUPPMCR4\_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

Generic system control

##### Reset value

See individual bit resets.

#### Bit descriptions

Figure A-27: AArch64\_imp\_cpuppmcr4\_el3 bit assignments

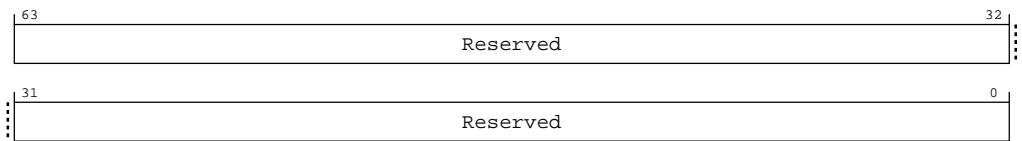


Table A-82: IMP\_CPUPPMCR4\_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

## Access

MRS <Xt>, S3\_6\_C15\_C2\_4

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_4	0b11	0b110	0b1111	0b0010	0b100

MSR S3\_6\_C15\_C2\_4, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_4	0b11	0b110	0b1111	0b0010	0b100

## Accessibility

MRS <Xt>, S3\_6\_C15\_C2\_4

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR4_EL3;

```

MSR S3\_6\_C15\_C2\_4, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPPMCR4_EL3 = X[t];

```

## A.1.28 IMP\_CPUPPMCR5\_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

Generic system control

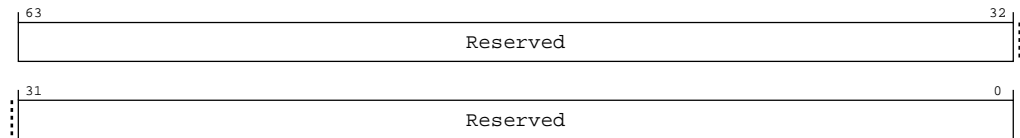


## Reset value

See individual bit resets.

## Bit descriptions

**Figure A-28: AArch64\_imp\_cpuppmcr5\_el3 bit assignments**



**Table A-85: IMP\_CPUPPMCR5\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

## Access

MRS <Xt>, S3\_6\_C15\_C2\_5

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_5	0b11	0b110	0b1111	0b0010	0b101

MSR S3\_6\_C15\_C2\_5, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_5	0b11	0b110	0b1111	0b0010	0b101

## Accessibility

MRS <Xt>, S3\_6\_C15\_C2\_5

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR5_EL3;

```

MSR S3\_6\_C15\_C2\_5, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR5_EL3 = X[t];

```

## A.1.29 IMP\_CPUPPMCR6\_EL3, CPU Power Performance Management Control Register

This register contains control bits that affect the CPU behavior

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

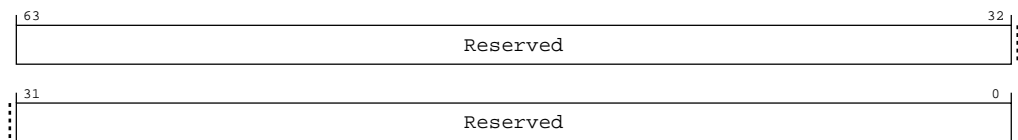
Generic system control

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-29: AArch64\_imp\_cpuppmcr6\_el3 bit assignments**



**Table A-88: IMP\_CPUPPMCR6\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

### Access

MRS <Xt>, S3\_6\_C15\_C2\_6

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_6	0b11	0b110	0b1111	0b0010	0b110

MSR S3\_6\_C15\_C2\_6, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C2_6	0b11	0b110	0b1111	0b0010	0b110

## Accessibility

MRS <Xt>, S3\_6\_C15\_C2\_6

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUPPMCR6_EL3;

```

MSR S3\_6\_C15\_C2\_6, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR6_EL3 = X[t];

```

## A.1.30 IMP\_CPUACTLR\_EL3, CPU Auxiliary Control Register (EL3)

This register contains control bits that affect the CPU behavior

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

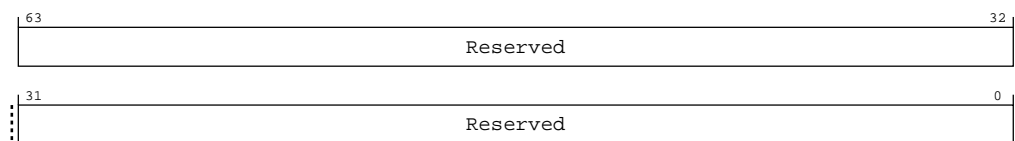
Generic system control

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-30: AArch64\_imp\_cpuactlr\_el3 bit assignments**



**Table A-91: IMP\_CPUACTLR\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

**Access**

MRS &lt;Xt&gt;, S3\_6\_C15\_C4\_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C4_0	0b11	0b110	0b1111	0b0100	0b000

MSR S3\_6\_C15\_C4\_0, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C4_0	0b11	0b110	0b1111	0b0100	0b000

**Accessibility**

MRS &lt;Xt&gt;, S3\_6\_C15\_C4\_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUACTLR_EL3;

```

MSR S3\_6\_C15\_C4\_0, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUACTLR_EL3 = X[t];

```

**A.1.31 IMP\_ATCR\_EL3, CPU Auxiliary Translation Control Register (EL2)**

This register contains control bits that affect the CPU behavior.

**Configurations**

This register is available in all configurations.

**Attributes****Width**

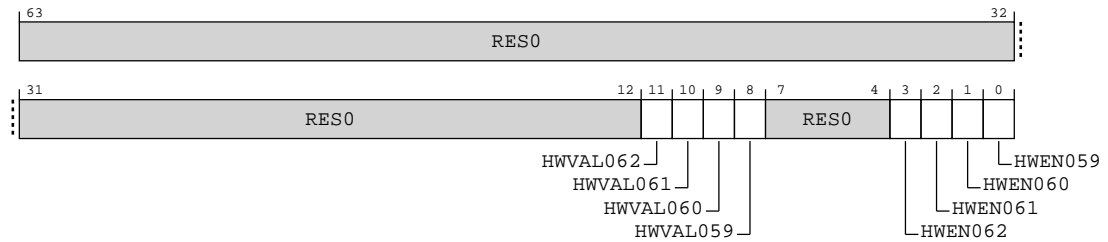
64

**Functional group**

Generic system control

**Reset value**

0x0

**Bit descriptions****Figure A-31: AArch64\_imp\_atcr\_el3 bit assignments****Table A-94: IMP\_ATCR\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:12]	RES0	Reserved	0x0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN062 is set.	0x0
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN061 is set.	0x0
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN060 is set.	0x0
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL3 if HWEN059 is set.	0x0
[7:4]	RES0	Reserved	0b0000
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to translation table walks using TTBR0_EL3. If this bit is clear, PBHA[3] will be 0 on translation table walks.	0b0
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to translation table walks using TTBR0_EL3. If this bit is clear, PBHA[2] will be 0 on translation table walks.	0b0
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to translation table walks using TTBR0_EL3. If this bit is clear, PBHA[1] will be 0 on translation table walks.	0b0
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to translation table walks using TTBR0_EL3. If this bit is clear, PBHA[0] will be 0 on translation table walks.	0b0

**Access**

MRS &lt;Xt&gt;, S3\_6\_C15\_C7\_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C7_0	0b11	0b110	0b1111	0b0111	0b000

MSR S3\_6\_C15\_C7\_0, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C7_0	0b11	0b110	0b1111	0b0111	0b000

### Accessibility

MRS <Xt>, S3\_6\_C15\_C7\_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_ATCR_EL3;
```

MSR S3\_6\_C15\_C7\_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_ATCR_EL3 = X[t];
```

## A.1.32 IMP\_CPUPSELR\_EL3, Selected Instruction Private Select Register

Selects the current instruction patch register for subsequent accesses to AArch64-IMP\_CPUPCR\_EL3, AArch64-IMP\_CPUPOR\_EL3, AArch64-IMP\_CPUPMR\_EL3, AArch64-IMP\_CPUPOR2\_EL3, AArch64-IMP\_CPUPMR2\_EL3, and AArch64-IMP\_CPUPFR\_EL3

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

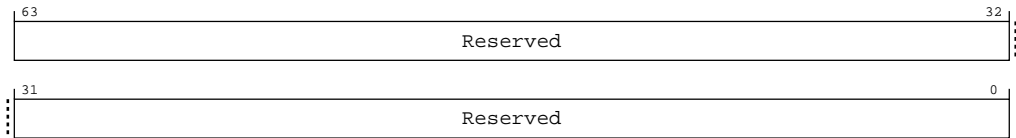
Generic system control

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-32: AArch64\_imp\_cpupselr\_el3 bit assignments**



**Table A-97: IMP\_CPUPSELR\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

## Access

MRS <Xt>, S3\_6\_C15\_C8\_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_0	0b11	0b110	0b1111	0b1000	0b000

MSR S3\_6\_C15\_C8\_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_0	0b11	0b110	0b1111	0b1000	0b000

## Accessibility

MRS <Xt>, S3\_6\_C15\_C8\_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPSELR_EL3;

```

MSR S3\_6\_C15\_C8\_0, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPSELR_EL3 = X[t];

```

### A.1.33 IMP\_CPUPCR\_EL3, Selected Instruction Private Control Register

Configures current Instruction Patch selected by AArch64-IMP\_CPUPSEL\_EL3.SEL.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

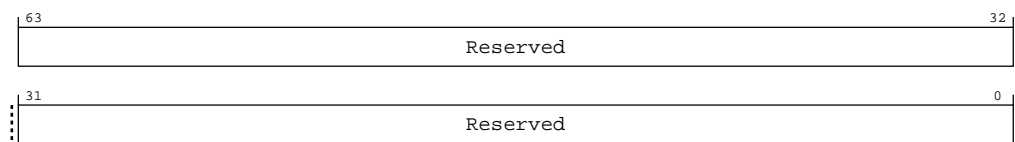
Generic system control

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure A-33: AArch64\_imp\_cpupcr\_el3 bit assignments**



**Table A-100: IMP\_CPUPCR\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

#### Access

MRS <Xt>, S3\_6\_C15\_C8\_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_1	0b11	0b110	0b1111	0b1000	0b001

MSR S3\_6\_C15\_C8\_1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_1	0b11	0b110	0b1111	0b1000	0b001

#### Accessibility

MRS <Xt>, S3\_6\_C15\_C8\_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then

```



```
    UNDEFINED;
  elseif PSTATE.EL == EL2 then
    UNDEFINED;
  elseif PSTATE.EL == EL3 then
    return IMP_CPUPCR_EL3;
```

MSR S3\_6\_C15\_C8\_1, <Xt>

```
  if PSTATE.EL == EL0 then
    UNDEFINED;
  elseif PSTATE.EL == EL1 then
    UNDEFINED;
  elseif PSTATE.EL == EL2 then
    UNDEFINED;
  elseif PSTATE.EL == EL3 then
    IMP_CPUPCR_EL3 = X[t];
```

A.1.34 IMP\_CPUPOR\_EL3, Selected Instruction Private Opcode Register

Opcode for current Instruction Patch selected by AArch64-IMP\_CPUPSELR\_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets.

Bit descriptions

Figure A-34: AArch64\_imp\_cpupor\_el3 bit assignments

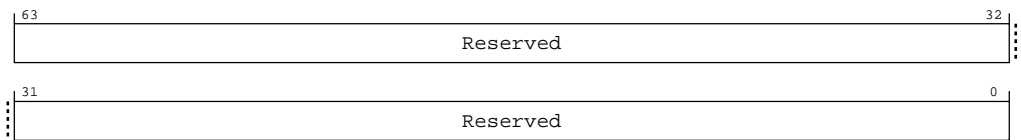


Table A-103: IMP\_CPUPOR\_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3\_6\_C15\_C8\_2

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_2	0b11	0b110	0b1111	0b1000	0b010

MSR S3\_6\_C15\_C8\_2, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_2	0b11	0b110	0b1111	0b1000	0b010

### Accessibility

MRS &lt;Xt&gt;, S3\_6\_C15\_C8\_2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPOR_EL3;

```

MSR S3\_6\_C15\_C8\_2, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPOR_EL3 = X[t];

```

## A.1.35 IMP\_CPUPMR\_EL3, Selected Instruction Private Mask Register

Mask for current Instruction Patch selected by AArch64-IMP\_CPUPSELR\_EL3.SEL.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

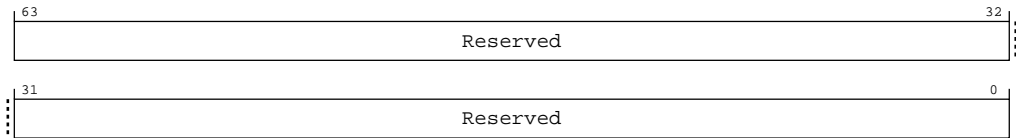
Generic system control

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-35: AArch64\_imp\_cpupmr\_el3 bit assignments**



**Table A-106: IMP\_CPUPMR\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

## Access

MRS <Xt>, S3\_6\_C15\_C8\_3

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_3	0b11	0b110	0b1111	0b1000	0b011

MSR S3\_6\_C15\_C8\_3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_3	0b11	0b110	0b1111	0b1000	0b011

## Accessibility

MRS <Xt>, S3\_6\_C15\_C8\_3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPMR_EL3;

```

MSR S3\_6\_C15\_C8\_3, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPMR_EL3 = X[t];

```

## A.1.36 IMP\_CPUPOR2\_EL3, Selected Instruction Private Opcode Register 2

Opcode exclusion for current Instruction Patch selected by AArch64-IMP\_CPUPSELR\_EL3.SEL.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

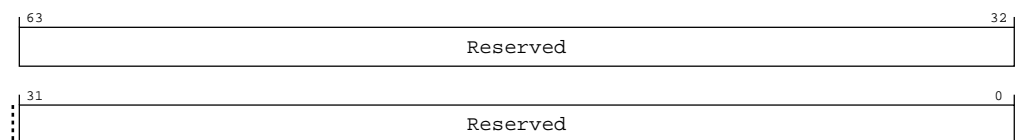
Generic system control

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-36: AArch64\_imp\_cpupor2\_el3 bit assignments**



**Table A-109: IMP\_CPUPOR2\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

### Access

MRS <Xt>, S3\_6\_C15\_C8\_4

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_4	0b11	0b110	0b1111	0b1000	0b100

MSR S3\_6\_C15\_C8\_4, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_4	0b11	0b110	0b1111	0b1000	0b100

### Accessibility

MRS <Xt>, S3\_6\_C15\_C8\_4

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then

```

```
    UNDEFINED;
  elseif PSTATE.EL == EL2 then
    UNDEFINED;
  elseif PSTATE.EL == EL3 then
    return IMP_CPUPOR2_EL3;
```

MSR S3\_6\_C15\_C8\_4, <Xt>

```
  if PSTATE.EL == EL0 then
    UNDEFINED;
  elseif PSTATE.EL == EL1 then
    UNDEFINED;
  elseif PSTATE.EL == EL2 then
    UNDEFINED;
  elseif PSTATE.EL == EL3 then
    IMP_CPUPOR2_EL3 = X[t];
```

A.1.37 IMP\_CPUPMR2\_EL3, Selected Instruction Private Mask Register 2

Mask exclusion for current Instruction Patch selected by AArch64-IMP\_CPUPSELR\_EL3.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

See individual bit resets.

Bit descriptions

Figure A-37: AArch64\_imp\_cpupmr2\_el3 bit assignments

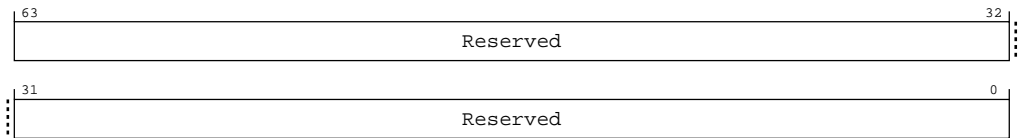


Table A-112: IMP\_CPUPMR2\_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

Access

MRS <Xt>, S3\_6\_C15\_C8\_5

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_5	0b11	0b110	0b1111	0b1000	0b101

MSR S3\_6\_C15\_C8\_5, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_5	0b11	0b110	0b1111	0b1000	0b101

### Accessibility

MRS &lt;Xt&gt;, S3\_6\_C15\_C8\_5

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPMR2_EL3;

```

MSR S3\_6\_C15\_C8\_5, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPMR2_EL3 = X[t];

```

## A.1.38 IMP\_CPUPFR\_EL3, Selected Instruction Private Flag Register

Instruction Patch flags for current Instruction Patch selected by AArch64-IMP\_CPUPSELR\_EL3.SEL.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

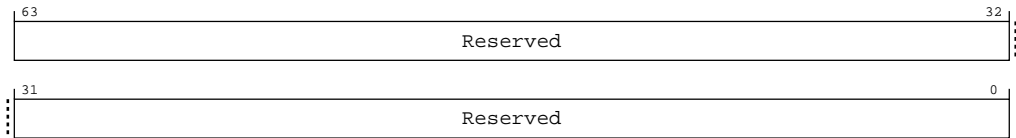
Generic system control

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-38: AArch64\_imp\_cpupfr\_el3 bit assignments**



**Table A-115: IMP\_CPUPFR\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use.	

## Access

MRS <Xt>, S3\_6\_C15\_C8\_6

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_6	0b11	0b110	0b1111	0b1000	0b110

MSR S3\_6\_C15\_C8\_6, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C8_6	0b11	0b110	0b1111	0b1000	0b110

## Accessibility

MRS <Xt>, S3\_6\_C15\_C8\_6

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_CPUPFR_EL3;

```

MSR S3\_6\_C15\_C8\_6, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    IMP_CPUPFR_EL3 = X[t];

```

### A.1.39 FPCR, Floating-point Control Register

Controls floating-point behavior.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

Generic system control

##### Reset value

See individual bit resets.

#### Bit descriptions

Figure A-39: AArch64\_fpcr bit assignments

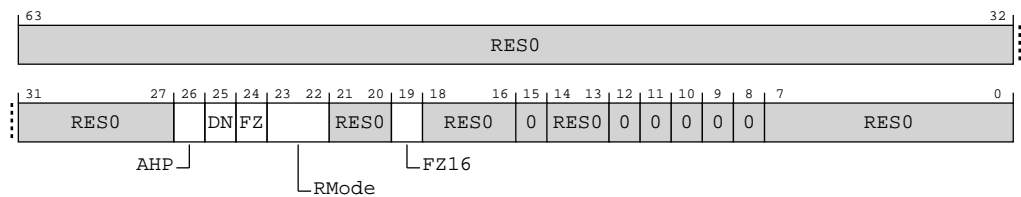


Table A-118: FPCR bit descriptions

Bits	Name	Description	Reset
[63:27]	RES0	Reserved	0x0
[26]	AHP	Alternative half-precision control bit:  <b>0b0</b> IEEE half-precision format selected.  <b>0b1</b> 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.	



Bits	Name	Description	Reset
[25]	DN	Default NaN mode control bit:  <b>0b0</b> NaN operands propagate through to the output of a floating-point operation.  <b>0b1</b> 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.	
[24]	FZ	Flush-to-zero mode control bit.  <b>0b0</b> Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.  <b>0b1</b> 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.	
[23:22]	RMode	Rounding Mode control field.  <b>0b00</b> Round to Nearest (RN) mode.  <b>0b01</b> Round towards Plus Infinity (RP) mode.  <b>0b10</b> Round towards Minus Infinity (RM) mode.  <b>0b11</b> Round towards Zero (RZ) mode.  The specified rounding mode is used by both scalar and Advanced SIMD floating-point instructions.	
[21:20]	RES0	Reserved	0b00
[19]	FZ16	Flush-to-zero mode control bit on half-precision data-processing instructions.  <b>0b0</b> Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard.  <b>0b1</b> 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.	
[18:0]	RES0	Reserved	0b00000000

## Access

MRS <Xt>, FPCR

<systemreg>	op0	op1	CRn	CRm	op2
FPCR	0b11	0b011	0b0100	0b0100	0b000

MSR FPCR, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
FPCR	0b11	0b011	0b0100	0b0100	0b000

## Accessibility

MRS &lt;Xt&gt;, FPCR

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.FPEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        return FPCR;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elseif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        return FPCR;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '0' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        return FPCR;
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        return FPCR;

```

MSR FPCR, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && CPTR_EL2.FPEN != '11' then

```

```

        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t];
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elseif EL2Enabled() && HCR_EL2.E2H != '1' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '0' && CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif HCR_EL2.E2H == '1' && CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elseif CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t];
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t];

```

### A.1.40 AFSR0\_EL2, Auxiliary Fault Status Register 0 (EL2)

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

#### Configurations

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

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

#### Attributes

##### Width

64

##### Functional group

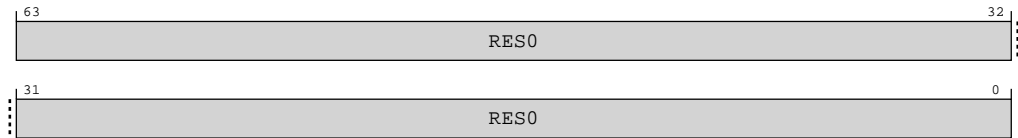
Generic system control

##### Reset value

0x0

## Bit descriptions

**Figure A-40: AArch64\_afsr0\_el2 bit assignments**



**Table A-121: AFSR0\_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

### Access

MRS <Xt>, AFSR0\_EL2

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL2	0b11	0b100	0b0101	0b0001	0b000

MSR AFSR0\_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL2	0b11	0b100	0b0101	0b0001	0b000

MRS <Xt>, AFSR0\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL1	0b11	0b000	0b0101	0b0001	0b000

MSR AFSR0\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL1	0b11	0b000	0b0101	0b0001	0b000

### Accessibility

MRS <Xt>, AFSR0\_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return AFSR0_EL2;
elsif PSTATE.EL == EL3 then
    return AFSR0_EL2;

```

## MSR AFSRO\_EL2, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    AFSRO_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    AFSRO_EL2 = X[t];

```

## MRS &lt;Xt&gt;, AFSRO\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AFSRO_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x128];
    else
        return AFSRO_EL1;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSRO_EL2;
    else
        return AFSRO_EL1;
elseif PSTATE.EL == EL3 then
    return AFSRO_EL1;

```

## MSR AFSRO\_EL1, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSRO_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x128] = X[t];
    else
        AFSRO_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        AFSRO_EL2 = X[t];
    else
        AFSRO_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    AFSRO_EL1 = X[t];

```

A.1.41 AFSR1\_EL2, Auxiliary Fault Status Register 1 (EL2)

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

Configurations

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

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

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-41: AArch64\_afsr1\_el2 bit assignments

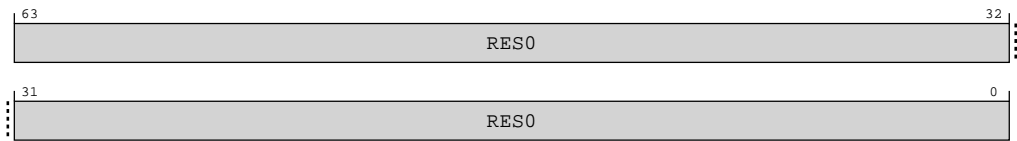


Table A-126: AFSR1\_EL2 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

Access

MRS <Xt>, AFSR1\_EL2

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL2	0b11	0b100	0b0101	0b0001	0b001

MSR AFSR1\_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL2	0b11	0b100	0b0101	0b0001	0b001

MRS <Xt>, AFSR1\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL1	0b11	0b000	0b0101	0b0001	0b001

MSR AFSR1\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL1	0b11	0b000	0b0101	0b0001	0b001

## Accessibility

MRS <Xt>, AFSR1\_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    return AFSR1_EL2;
elseif PSTATE.EL == EL3 then
    return AFSR1_EL2;

```

MSR AFSR1\_EL2, <Xt>

```

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

```

MRS <Xt>, AFSR1\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x130];
    else
        return AFSR1_EL1;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR1_EL2;
    else
        return AFSR1_EL1;
elseif PSTATE.EL == EL3 then
    return AFSR1_EL1;

```

MSR AFSR1\_EL1, <Xt>

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

A.1.42 AFSR0\_EL1, Auxiliary Fault Status Register 0 (EL1)

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure A-42: AArch64\_afsr0\_el1 bit assignments

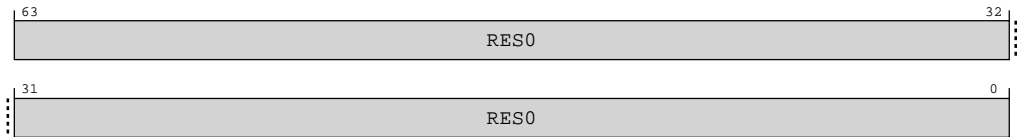


Table A-131: AFSR0\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0



## Access

MRS <Xt>, AFSRO\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AFSRO_EL1	0b11	0b000	0b0101	0b0001	0b000

MSR AFSRO\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSRO_EL1	0b11	0b000	0b0101	0b0001	0b000

MRS <Xt>, AFSRO\_EL12

<systemreg>	op0	op1	CRn	CRm	op2
AFSRO_EL12	0b11	0b101	0b0101	0b0001	0b000

MSR AFSRO\_EL12, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSRO_EL12	0b11	0b101	0b0101	0b0001	0b000

## Accessibility

MRS <Xt>, AFSRO\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AFSRO_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x128];
    else
        return AFSRO_EL1;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSRO_EL2;
    else
        return AFSRO_EL1;
elseif PSTATE.EL == EL3 then
    return AFSRO_EL1;

```

MSR AFSRO\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.AFSRO_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        NVMem[0x128] = X[t];
    else

```

```

        AFSR0_EL1 = X[t];
    elseif PSTATE.EL == EL2 then
        if HCR_EL2.E2H == '1' then
            AFSR0_EL2 = X[t];
        else
            AFSR0_EL1 = X[t];
    elseif PSTATE.EL == EL3 then
        AFSR0_EL1 = X[t];

```

MRS <Xt>, AFSR0\_EL12

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        return NVMem[0x128];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR0_EL1;
    else
        UNDEFINED;
elseif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        return AFSR0_EL1;
    else
        UNDEFINED;

```

MSR AFSR0\_EL12, <Xt>

```

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

```

### A.1.43 AFSR1\_EL1, Auxiliary Fault Status Register 1 (EL1)

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

#### Configurations

This register is available in all configurations.

**Attributes****Width**

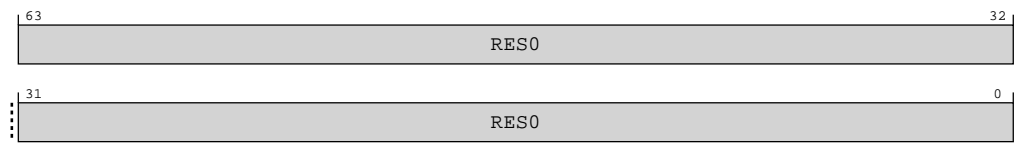
64

**Functional group**

Generic system control

**Reset value**

0x0

**Bit descriptions****Figure A-43: AArch64\_afsr1\_el1 bit assignments****Table A-136: AFSR1\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

**Access**

MRS &lt;Xt&gt;, AFSR1\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL1	0b11	0b000	0b0101	0b0001	0b001

MSR AFSR1\_EL1, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL1	0b11	0b000	0b0101	0b0001	0b001

MRS &lt;Xt&gt;, AFSR1\_EL12

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL12	0b11	0b101	0b0101	0b0001	0b001

MSR AFSR1\_EL12, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL12	0b11	0b101	0b0101	0b0001	0b001

## Accessibility

MRS <Xt>, AFSR1\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.AFSR1_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '111' then
        return NVMem[0x130];
    else
        return AFSR1_EL1;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR1_EL2;
    else
        return AFSR1_EL1;
elseif PSTATE.EL == EL3 then
    return AFSR1_EL1;

```

MSR AFSR1\_EL1, <Xt>

```

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

```

MRS <Xt>, AFSR1\_EL12

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV1,NV> == '101' then
        return NVMem[0x130];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' then
        return AFSR1_EL1;
    else
        UNDEFINED;
elseif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == '1' then
        return AFSR1_EL1;
    else

```

```
UNDEFINED;
```

MSR AFSR1\_EL12, &lt;Xt&gt;

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

### A.1.44 AFSR0\_EL3, Auxiliary Fault Status Register 0 (EL3)

Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL3.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

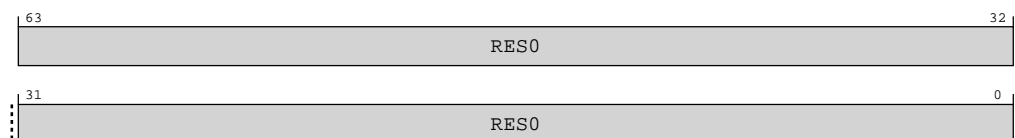
Generic system control

##### Reset value

0x0

#### Bit descriptions

**Figure A-44: AArch64\_afsr0\_el3 bit assignments**



**Table A-141: AFSRO\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

**Access**

MRS &lt;Xt&gt;, AFSRO\_EL3

<systemreg>	op0	op1	CRn	CRm	op2
AFSRO_EL3	0b11	0b110	0b0101	0b0001	0b000

MSR AFSRO\_EL3, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
AFSRO_EL3	0b11	0b110	0b0101	0b0001	0b000

**Accessibility**

MRS &lt;Xt&gt;, AFSRO\_EL3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return AFSRO_EL3;

```

MSR AFSRO\_EL3, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    AFSRO_EL3 = X[t];

```

**A.1.45 AFSR1\_EL3, Auxiliary Fault Status Register 1 (EL3)**Provides additional **IMPLEMENTATION DEFINED** fault status information for exceptions taken to EL3.**Configurations**

This register is available in all configurations.

**Attributes****Width**

64

**Functional group**  
Generic system control

**Reset value**  
0x0

Bit descriptions

Figure A-45: AArch64\_afsr1\_el3 bit assignments

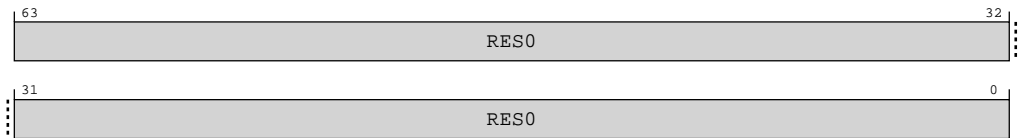


Table A-144: AFSR1\_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

**Access**  
MRS <Xt>, AFSR1\_EL3

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL3	0b11	0b110	0b0101	0b0001	0b001

MSR AFSR1\_EL3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL3	0b11	0b110	0b0101	0b0001	0b001

**Accessibility**  
MRS <Xt>, AFSR1\_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return AFSR1_EL3;
```

MSR AFSR1\_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
```

```
elseif PSTATE.EL == EL3 then
    AFSR1_EL3 = X[t];
```

## A.2 AArch64 Debug register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Debug registers in the core. Individual register descriptions provide detailed information.

**Table A-147: Debug register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
IMP_IDATA0_EL3	3	C15	6	C0	0	See individual bit resets.	64-bit	Instruction Register 0
IMP_IDATA1_EL3	3	C15	6	C0	1	See individual bit resets.	64-bit	Instruction Register 0
IMP_IDATA2_EL3	3	C15	6	C0	2	See individual bit resets.	64-bit	Instruction Register 0
IMP_DDATA0_EL3	3	C15	6	C1	0	See individual bit resets.	64-bit	Data Register 0
IMP_DDATA1_EL3	3	C15	6	C1	1	See individual bit resets.	64-bit	Data Register 1
IMP_DDATA2_EL3	3	C15	6	C1	2	See individual bit resets.	64-bit	Data Register 2

### A.2.1 IMP\_IDATA0\_EL3, Instruction Register 0

Contains data from a preceding RAMINDEX operation.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

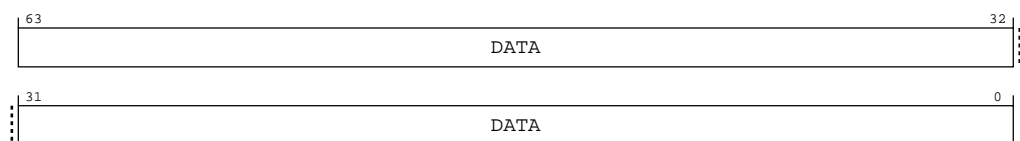
Debug

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure A-46: AArch64\_imp\_idata0\_el3 bit assignments**





**Table A-148: IMP\_IDATA0\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	

**Access**

MRS &lt;Xt&gt;, S3\_6\_C15\_C0\_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C0_0	0b11	0b110	0b1111	0b0000	0b000

**Accessibility**

MRS &lt;Xt&gt;, S3\_6\_C15\_C0\_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_IDATA0_EL3;

```

**A.2.2 IMP\_IDATA1\_EL3, Instruction Register 0**

Contains data from a preceding RAMINDEX operation.

**Configurations**

This register is available in all configurations.

**Attributes****Width**

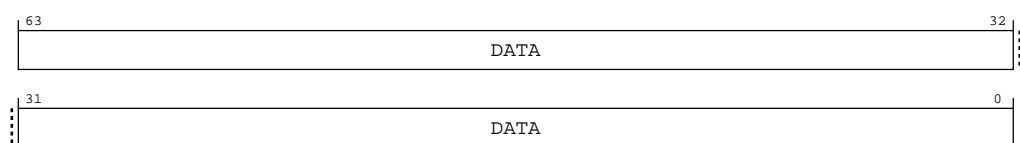
64

**Functional group**

Debug

**Reset value**

See individual bit resets.

**Bit descriptions****Figure A-47: AArch64\_imp\_idata1\_el3 bit assignments**

**Table A-150: IMP\_IDATA1\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	

**Access**

MRS &lt;Xt&gt;, S3\_6\_C15\_CO\_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_CO_1	0b11	0b110	0b1111	0b0000	0b001

**Accessibility**

MRS &lt;Xt&gt;, S3\_6\_C15\_CO\_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_IDATA1_EL3;

```

**A.2.3 IMP\_IDATA2\_EL3, Instruction Register 0**

Contains data from a preceding RAMINDEX operation.

**Configurations**

This register is available in all configurations.

**Attributes****Width**

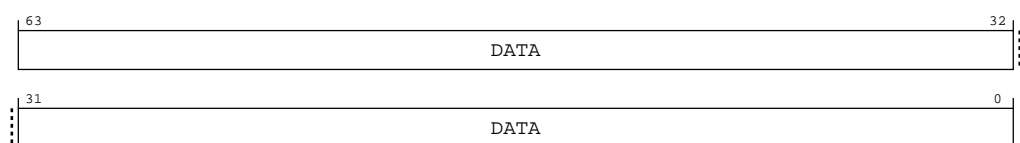
64

**Functional group**

Debug

**Reset value**

See individual bit resets.

**Bit descriptions****Figure A-48: AArch64\_imp\_idata2\_el3 bit assignments**

**Table A-152: IMP\_IDATA2\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	

**Access**

MRS &lt;Xt&gt;, S3\_6\_C15\_C0\_2

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C0_2	0b11	0b110	0b1111	0b0000	0b010

**Accessibility**

MRS &lt;Xt&gt;, S3\_6\_C15\_C0\_2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_IDATA2_EL3;

```

**A.2.4 IMP\_DDATA0\_EL3, Data Register 0**

Contains data from a preceding RAMINDEX operation.

**Configurations**

This register is available in all configurations.

**Attributes****Width**

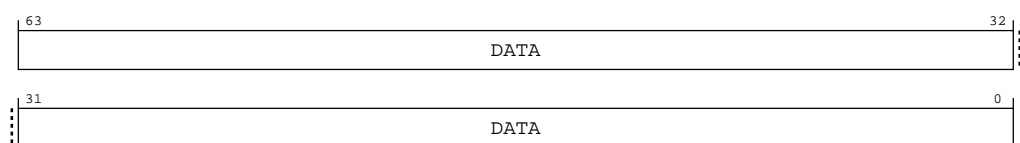
64

**Functional group**

Debug

**Reset value**

See individual bit resets.

**Bit descriptions****Figure A-49: AArch64\_imp\_ddata0\_el3 bit assignments**

**Table A-154: IMP\_DDATA0\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	

**Access**

MRS &lt;Xt&gt;, S3\_6\_C15\_C1\_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C1_0	0b11	0b110	0b1111	0b0001	0b000

**Accessibility**

MRS &lt;Xt&gt;, S3\_6\_C15\_C1\_0

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_DDATA0_EL3;

```

**A.2.5 IMP\_DDATA1\_EL3, Data Register 1**

Contains data from a preceding RAMINDEX operation.

**Configurations**

This register is available in all configurations.

**Attributes****Width**

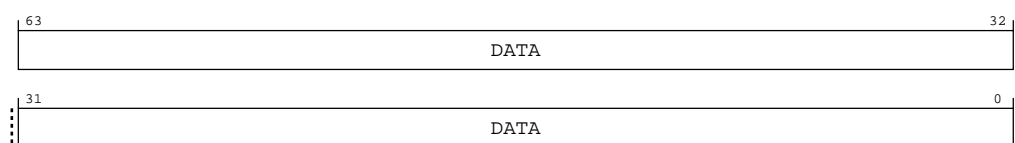
64

**Functional group**

Debug

**Reset value**

See individual bit resets.

**Bit descriptions****Figure A-50: AArch64\_imp\_ddata1\_el3 bit assignments**

**Table A-156: IMP\_DDATA1\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	

**Access**

MRS &lt;Xt&gt;, S3\_6\_C15\_C1\_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C1_1	0b11	0b110	0b1111	0b0001	0b001

**Accessibility**

MRS &lt;Xt&gt;, S3\_6\_C15\_C1\_1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_DDATA1_EL3;

```

**A.2.6 IMP\_DDATA2\_EL3, Data Register 2**

Contains data from a preceding RAMINDEX operation.

**Configurations**

This register is available in all configurations.

**Attributes****Width**

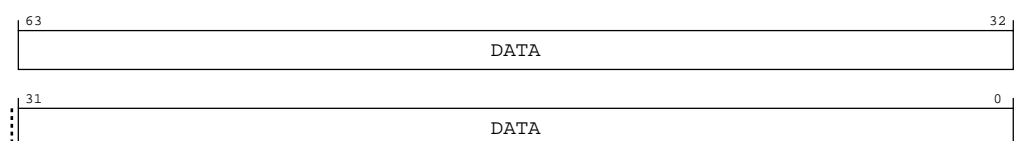
64

**Functional group**

Debug

**Reset value**

See individual bit resets.

**Bit descriptions****Figure A-51: AArch64\_imp\_ddata2\_el3 bit assignments**

**Table A-158: IMP\_DDATA2\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:0]	DATA	Contains data from a preceding RAMINDEX operation	

**Access**

MRS &lt;Xt&gt;, S3\_6\_C15\_C1\_2

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C1_2	0b11	0b110	0b1111	0b0001	0b010

**Accessibility**

MRS &lt;Xt&gt;, S3\_6\_C15\_C1\_2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return IMP_DDATA2_EL3;

```

## A.3 System instruction register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** System instruction registers in the core. Individual register descriptions provide detailed information.

**Table A-160: System instruction register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
SYS_IMP_RAMINDEX	1	C15	6	C0	0	See individual bit resets.	64-bit	RAM Index

### A.3.1 SYS\_IMP\_RAMINDEX, RAM Index

Read contents of the cache specified by the source register into AArch64-IMP\_IDATA0\_EL3, AArch64-IMP\_IDATA1\_EL3, AArch64-IMP\_IDATA2\_EL3, AArch64-IMP\_DDATA0\_EL3, AArch64-IMP\_DDATA1\_EL3, and AArch64-IMP\_DDATA2\_EL3.

**Configurations**

This register is available in all configurations.

**Attributes****Width**

64

Functional group

System instruction

Reset value

See individual bit resets.

Bit descriptions

Figure A-52: AArch64\_sys\_imp\_ramindex bit assignments

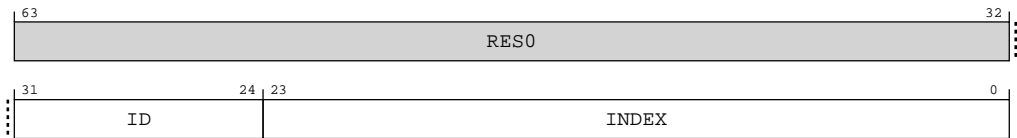


Table A-161: SYS\_IMP\_RAMINDEX bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:24]	ID	RAM ID (See Chapter 10)	
[23:0]	INDEX	RAM Index (See Chapter 10)	

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C0, #0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C0_0	0b01	0b110	0b1111	0b0000	0b000

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C0, #0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_RAMINDEX(X[t]);
```

## A.4 AArch64 Identification register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Identification registers in the core. Individual register descriptions provide detailed information.

**Table A-163: Identification register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
MIDR_EL1	3	C0	0	C0	0	See individual bit resets.	64-bit	Main ID Register
MPIDR_EL1	3	C0	0	C0	5	See individual bit resets.	64-bit	Multiprocessor Affinity Register
REVIDR_EL1	3	C0	0	C0	6	See individual bit resets.	64-bit	Revision ID Register
ID_AA64PFR0_EL1	3	C0	0	C4	0	See individual bit resets.	64-bit	AArch64 Processor Feature Register 0
ID_AA64PFR1_EL1	3	C0	0	C4	1	See individual bit resets.	64-bit	AArch64 Processor Feature Register 1
ID_AA64ZFR0_EL1	3	C0	0	C4	4	See individual bit resets.	64-bit	SVE Feature ID register 0
ID_AA64DFR0_EL1	3	C0	0	C5	0	See individual bit resets.	64-bit	AArch64 Debug Feature Register 0
ID_AA64DFR1_EL1	3	C0	0	C5	1	0x0	64-bit	AArch64 Debug Feature Register 1
ID_AA64AFR0_EL1	3	C0	0	C5	4	0x0	64-bit	AArch64 Auxiliary Feature Register 0
ID_AA64AFR1_EL1	3	C0	0	C5	5	0x0	64-bit	AArch64 Auxiliary Feature Register 1
ID_AA64ISAR0_EL1	3	C0	0	C6	0	See individual bit resets.	64-bit	AArch64 Instruction Set Attribute Register 0
ID_AA64ISAR1_EL1	3	C0	0	C6	1	See individual bit resets.	64-bit	AArch64 Instruction Set Attribute Register 1
ID_AA64MMFR0_EL1	3	C0	0	C7	0	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 0
ID_AA64MMFR1_EL1	3	C0	0	C7	1	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 1
ID_AA64MMFR2_EL1	3	C0	0	C7	2	See individual bit resets.	64-bit	AArch64 Memory Model Feature Register 2
CLIDR_EL1	3	C0	1	C0	1	See individual bit resets.	64-bit	Cache Level ID Register
GMIDR_EL1	3	C0	1	C0	4	See individual bit resets.	64-bit	Multiple tag transfer ID register
CTR_EL0	3	C0	3	C0	1	See individual bit resets.	64-bit	Cache Type Register
DCZID_EL0	3	C0	3	C0	7	See individual bit resets.	64-bit	Data Cache Zero ID register
MPAMIDR_EL1	3	C10	0	C4	4	See individual bit resets.	64-bit	MPAM ID Register (EL1)
IMP_CPUCFR_EL1	3	C15	0	C0	0	See individual bit resets.	64-bit	CPU Configuration Register

### A.4.1 MIDR\_EL1, Main ID Register

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

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

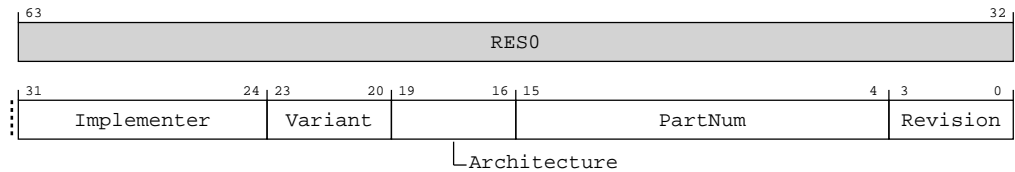
##### Functional group

identification



**Reset value**

See individual bit resets.

**Bit descriptions****Figure A-53: AArch64\_midr\_el1 bit assignments****Table A-164: MIDR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:24]	Implementer	Indicates the implementer code. This value is: <b>0b01000001</b> Arm Limited	
[23:20]	Variant	An <b>IMPLEMENTATION DEFINED</b> variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product. <b>0b0010</b> r2p1	
[19:16]	Architecture	Indicates the architecture code. This value is: <b>0b1111</b> Architecture is defined by ID registers	
[15:4]	PartNum	An <b>IMPLEMENTATION DEFINED</b> 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. <b>0b110101001000</b> Cortex-X2	
[3:0]	Revision	An <b>IMPLEMENTATION DEFINED</b> revision number for the device. <b>0b0001</b> r2p1	

**Access**

MRS &lt;Xt&gt;, MIDR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MIDR_EL1	0b11	0b000	0b0000	0b0000	0b000

## Accessibility

MRS <Xt>, MIDR\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.MIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() then
        return VPIDR_EL2;
    else
        return MIDR_EL1;
elseif PSTATE.EL == EL2 then
    return MIDR_EL1;
elseif PSTATE.EL == EL3 then
    return MIDR_EL1;

```

## A.4.2 MPIDR\_EL1, Multiprocessor Affinity Register

In a multiprocessor system, provides an additional PE identification mechanism for scheduling purposes.

### Configurations

In a uniprocessor system Arm recommends that each Aff<n> field of this register returns a value of 0.

### Attributes

#### Width

64

#### Functional group

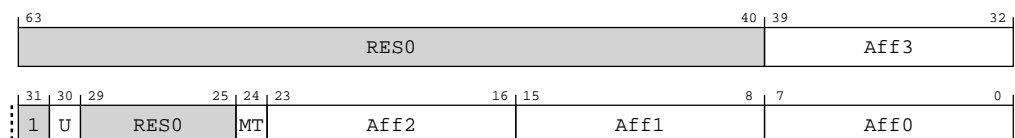
identification

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-54: AArch64\_mpidr\_el1 bit assignments**



**Table A-166: MPIDR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	0x0
[39:32]	Aff3	Affinity level 3. See the description of Aff0 for more information.  The value will be determined by the CLUSTERIDAFF3 configuration pins.	
[31]	RES1	Reserved	0b1
[30]	U	Indicates a Uniprocessor system, as distinct from PE 0 in a multiprocessor system. The possible values of this bit are:  <b>0b0</b> Processor is part of a multiprocessor system.	
[29:25]	RES0	Reserved	0b00000
[24]	MT	Indicates whether the lowest level of affinity consists of logical PEs that are implemented using a multithreading type approach. See the description of Aff0 for more information about affinity levels. The possible values of this bit are:  <b>0b1</b> Performance of PEs at the lowest affinity level, or PEs with MPIDR_EL1.MT set to 1, different affinity level 0 values, and the same values for affinity level 1 and higher, is very interdependent.	
[23:16]	Aff2	Affinity level 2. See the description of Aff0 for more information.  The value will be determined by the CLUSTERIDAFF2 configuration pins.	
[15:8]	Aff1	Affinity level 1. See the description of Aff0 for more information.  Value read from the CUID configuration pins. Identification number for each CPU in an cluster counting from zero.	
[7:0]	Aff0	Affinity level 0. This is the affinity level that is most significant for determining PE behavior. Higher affinity levels are increasingly less significant in determining PE behavior. The assigned value of the MPIDR.{Aff2, Aff1, Aff0} or AArch64-MPIDR_EL1.{Aff3, Aff2, Aff1, Aff0} set of fields of each PE must be unique within the system as a whole.  <b>0b00000000</b> Only one thread.	

### Access

MRS &lt;Xt&gt;, MPIDR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MPIDR_EL1	0b11	0b000	0b0000	0b0000	0b101

### Accessibility

MRS &lt;Xt&gt;, MPIDR\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.MPIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);

```

```

    elif EL2Enabled() then
        return VMPIDR_EL2;
    else
        return MPIDR_EL1;
    elif PSTATE.EL == EL2 then
        return MPIDR_EL1;
    elif PSTATE.EL == EL3 then
        return MPIDR_EL1;

```

### A.4.3 REVIDR\_EL1, Revision ID Register

The REVIDR\_EL1 provides revision information, additional to MIDR\_EL1, that identifies minor fixes (errata) which might be present in a specific implementation of the Cortex-X2 core. Refer to the Cortex-X2 Product Errata Notice (PEN) for information on how to interpret the values in this register.

#### Configurations

If REVIDR\_EL1 has the same value as AArch64-MIDR\_EL1, then its contents have no significance.

#### Attributes

##### Width

64

##### Functional group

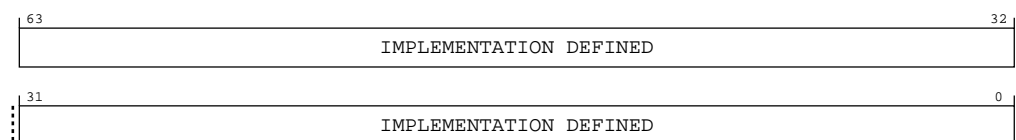
Identification

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure A-55: AArch64\_revidr\_el1 bit assignments**



**Table A-168: REVIDR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:0]	IMPLEMENTATION DEFINED	IMPLEMENTATION DEFINED	

#### Access

MRS <Xt>, REVIDR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
REVIDR_EL1	0b11	0b000	0b0000	0b0000	0b110

## Accessibility

MRS <Xt>, REVIDR\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.REVIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return REVIDR_EL1;
elseif PSTATE.EL == EL2 then
    return REVIDR_EL1;
elseif PSTATE.EL == EL3 then
    return REVIDR_EL1;

```

## A.4.4 ID\_AA64PFR0\_EL1, AArch64 Processor Feature Register 0

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'.

### Configurations

The external register ext-EDPFR gives information from this register.

### Attributes

#### Width

64

#### Functional group

Identification

#### Reset value

See individual bit resets.

### Bit descriptions

Figure A-56: AArch64\_id\_aa64pfr0\_el1 bit assignments

63	60	59	56	55	52	51	48	47	44	43	40	39	36	35	32
CSV3	CSV2	RES0	DIT	AMU	MPAM	SEL2	SVE								
31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0
RAS	GIC	AdvSIMD	FP	EL3	EL2	EL1	EL0								

**Table A-170: ID\_AA64PFR0\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:60]	CSV3	Speculative use of faulting data. Defined values are: <b>0b0001</b> Data loaded under speculation with a permission or domain fault cannot be used to form an address or generate condition codes or SVE predicate values to be used by instructions newer than the load in the speculative sequence	
[59:56]	CSV2	Speculative use of out of context branch targets. Defined values are: <b>0b0010</b> Branch targets trained in one hardware described context can only affect speculative execution in a different hardware described context in a hard-to-determine way. Contexts include the SCXTNUM_ELx register contexts, and these registers are supported.	
[55:52]	RES0	Reserved	0b0000
[51:48]	DIT	Data Independent Timing. Defined values are: <b>0b0001</b> AArch64 provides the PSTATE.DIT mechanism to guarantee constant execution time of certain instructions.	
[47:44]	AMU	Indicates support for Activity Monitors Extension. Defined values are: <b>0b0001</b> AMUv1 for Armv8.4 is implemented.	
[43:40]	MPAM	Indicates support for MPAM Extension. Defined values are: <b>0b0001</b> If AArch64-ID_AA64PFR1_EL1.MPAM_frac == 0b0000, MPAM Extension version 1.0 is implemented.  If AArch64-ID_AA64PFR1_EL1.MPAM_frac == 0b0001, MPAM Extension version 1.1 is implemented.	
[39:36]	SEL2	Secure EL2. Defined values are: <b>0b0001</b> Secure EL2 is implemented.	
[35:32]	SVE	Scalable Vector Extension. Defined values are: <b>0b0001</b> SVE architectural state and programmers' model are implemented.	
[31:28]	RAS	RAS Extension version. Defined values are: <b>0b0010</b> ARMv8.4-RAS present. As 0b0001, and adds support for ARMv8.4-DFE (If EL3 is implemented), additional ERXMISCM_EL1 System registers, additionalSystem registers ERXPFGCDN_EL1, ERXPFGCTL_EL1, and ERXPFGF_EL1, and the SCR_EL3.FIEN and HCR_EL2.FIEN trap controls, to support the optional RAS Common Fault Injection Model Extension.	
[27:24]	GIC	System register GIC CPU interface. Defined values are: <b>0b0000</b> When Port GICCDISABLE is High, GIC CPU interface is disabled. <b>0b0011</b> When Port GICCDISABLE is Low, GIC (version 4.1) CPU interface is enabled.	

Bits	Name	Description	Reset
[23:20]	AdvSIMD	Advanced SIMD. Defined values are:  <b>0b0001</b> Advanced SIMD is implemented, including support for half-precision floating-point arithmetic.	
[19:16]	FP	Floating-point. Defined values are:  <b>0b0001</b> Floating-point, including support for half-precision floating-point arithmetic, is implemented.	
[15:12]	EL3	EL3 Exception level handling. Defined values are:  <b>0b0001</b> EL3 can be executed in AArch64 state only.	
[11:8]	EL2	EL2 Exception level handling. Defined values are:  <b>0b0001</b> EL2 can be executed in AArch64 state only.	
[7:4]	EL1	EL1 Exception level handling. Defined values are:  <b>0b0001</b> EL1 can be executed in AArch64 state only.	
[3:0]	ELO	ELO Exception level handling. Defined values are:  <b>0b0001</b> ELO can be executed in AArch64 state only.	

## Access

MRS <Xt>, ID\_AA64PFR0\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64PFR0_EL1	0b11	0b000	0b0000	0b0100	0b000

## Accessibility

MRS <Xt>, ID\_AA64PFR0\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64PFR0_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64PFR0_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64PFR0_EL1;

```

A.4.5 ID\_AA64PFR1\_EL1, AArch64 Processor Feature Register 1

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'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure A-57: AArch64\_id\_aa64pfr1\_el1 bit assignments

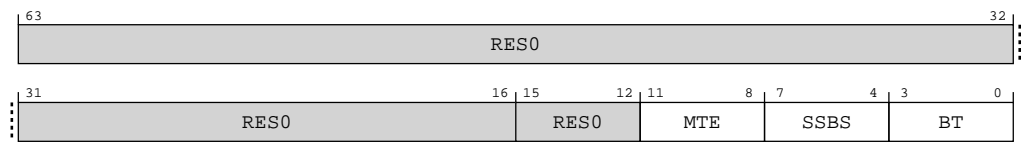


Table A-172: ID\_AA64PFR1\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:12]	RES0	Reserved	0b0000
[11:8]	MTE	Support for the Memory Tagging Extension. Defined values are:  <b>0b0001</b> Memory Tagging Extension instructions accessible at EL0 are implemented. Instructions and System Registers defined by the extension not configurably accessible at EL0 are Unallocated and other System Register fields defined by the extension are RES0. This value is reported when the BROADCASTMTE input is LOW.  <b>0b0010</b> Memory Tagging Extension is implemented. This value is reported when the BROADCASTMTE input is HIGH.	
[7:4]	SSBS	Speculative Store Bypassing controls in AArch64 state. Defined values are:  <b>0b0010</b> AArch64 provides the PSTATE.SSBS mechanism to mark regions that are Speculative Store Bypassing Safe, and the MSR and MRS instructions to directly read and write the PSTATE.SSBS field	



Bits	Name	Description	Reset
[3:0]	BT	Branch Target Identification mechanism support in AArch64 state. Defined values are:  <b>0b0001</b> The Branch Target Identification mechanism is implemented.	

### Access

MRS &lt;Xt&gt;, ID\_AA64PFR1\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64PFR1_EL1	0b11	0b000	0b0000	0b0100	0b001

### Accessibility

MRS &lt;Xt&gt;, ID\_AA64PFR1\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64PFR1_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64PFR1_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64PFR1_EL1;

```

## A.4.6 ID\_AA64ZFR0\_EL1, SVE Feature ID register 0

Provides additional information about the implemented features of the AArch64 Scalable Vector Extension, when the AArch64-ID\_AA64PFR0\_EL1.SVE field is not zero.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

### Configurations

**Note**

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

### Attributes

#### Width

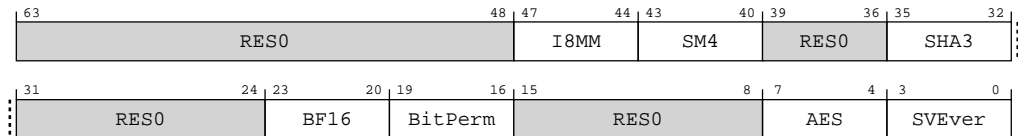
64

**Functional group**

Identification

**Reset value**

See individual bit resets.

**Bit descriptions****Figure A-58: AArch64\_id\_aa64zfr0\_el1 bit assignments****Table A-174: ID\_AA64ZFR0\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	0x0
[47:44]	I8MM	Indicates support for SVE Int8 matrix multiplication instructions. Defined values are: <b>0b0001</b> SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.	
[43:40]	SM4	Indicates support for SVE2 SM4 instructions. Defined values are: <b>0b0000</b> SVE2 SM4 instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. <b>0b0001</b> SVE2 SM4E and SM4EKEY instructions are implemented. This value is reported when Cryptographic extensions are implemented and enabled.	
[39:36]	RES0	Reserved	0b0000
[35:32]	SHA3	Indicates support for the SVE2 SHA-3 instruction. Defined values are: <b>0b0000</b> SVE2 SHA-3 instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. <b>0b0001</b> SVE2 RAX1 instruction is implemented. This value is reported when Cryptographic extensions are implemented and enabled.	
[31:24]	RES0	Reserved	0b00000000
[23:20]	BF16	Indicates support for SVE BFloat16 instructions. Defined values are: <b>0b0001</b> BFCVT, BFCVTNT, BFDOT, BFMLALB, BFMLALT, and BFMMMLA instructions are implemented.	
[19:16]	BitPerm	Indicates support for SVE2 bit permute instructions. Defined values are: <b>0b0001</b> SVE2 BDEP, BEXT and BGRP instructions are implemented.	
[15:8]	RES0	Reserved	0b00000000

Bits	Name	Description	Reset
[7:4]	AES	Indicates support for SVE2-AES instructions. Defined values are:  <b>0b0000</b> SVE2-AES instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled.  <b>0b0010</b> SVE2 AESE, AESD, AESMC, and AESIMC instructions are implemented plus SVE2 PMULLB and PMULLT instructions with 64-bit source. This value is reported when Cryptographic extensions are implemented and enabled.	
[3:0]	SVEver	Scalable Vector Extension instruction set version. Defined values are:  <b>0b0001</b> SVE and the non-optional SVE2 instructions are implemented.	

### Access

MRS <Xt>, ID\_AA64ZFR0\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64ZFR0_EL1	0b11	0b000	0b0000	0b0100	0b100

### Accessibility

MRS <Xt>, ID\_AA64ZFR0\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && (!IsZero(ID_AA64ZFR0_EL1) || boolean IMPLEMENTATION_DEFINED
        "ID_AA64ZFR0_EL1 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64ZFR0_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64ZFR0_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64ZFR0_EL1;

```

## A.4.7 ID\_AA64DFR0\_EL1, AArch64 Debug Feature Register 0

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.

### Configurations

The external register ext-EDDFR gives information from this register.

## Attributes

### Width

64

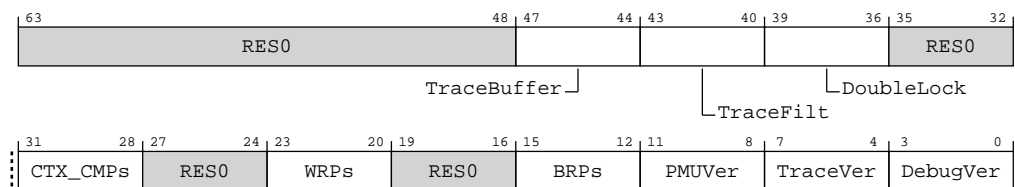
### Functional group

Identification

### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-59: AArch64\_id\_aa64dfr0\_el1 bit assignments****Table A-176: ID\_AA64DFR0\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	0x0
[47:44]	TraceBuffer	Trace Buffer Extension version. Defined values are: <b>0b0001</b> Trace Buffer Extension implemented.	
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. Defined values are: <b>0b0001</b> Armv8.4 Self-hosted Trace Extension implemented.	
[39:36]	DoubleLock	OS Double Lock implemented. Defined values are: <b>0b1111</b> OS Double Lock not implemented. AArch64-OSDLR_EL1 is RAZ/WI.	
[35:32]	RES0	Reserved	0b0000
[31:28]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. <b>0b0001</b> Two context-aware breakpoints are included	
[27:24]	RES0	Reserved	0b0000
[23:20]	WRPs	Number of watchpoints, minus 1. The value of 0b0000 is reserved. <b>0b0011</b> Four Watchpoints	
[19:16]	RES0	Reserved	0b0000
[15:12]	BRPs	Number of breakpoints, minus 1. The value of 0b0000 is reserved. <b>0b0101</b> Six Breakpoints	

Bits	Name	Description	Reset
[11:8]	PMUVer	Performance Monitors Extension version. Defined value is:  <b>0b0110</b> Performance Monitors Extension implemented, PMUv3 for Armv8.5	
[7:4]	TraceVer	Trace support. Indicates whether System register interface to a PE Trace unit is implemented. Defined values are:  <b>0b0001</b> PE Trace unit System registers implemented.	
[3:0]	DebugVer	Debug architecture version. Indicates presence of Armv8 Debug architecture. Defined values are:  <b>0b1001</b> Armv8.4 Debug architecture.	

### Access

MRS <Xt>, ID\_AA64DFR0\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64DFR0_EL1	0b11	0b000	0b0000	0b0101	0b000

### Accessibility

MRS <Xt>, ID\_AA64DFR0\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64DFR0_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64DFR0_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64DFR0_EL1;

```

## A.4.8 ID\_AA64DFR1\_EL1, AArch64 Debug Feature Register 1

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'.

### Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

0x0

Bit descriptions

Figure A-60: AArch64\_id\_aa64dfr1\_el1 bit assignments

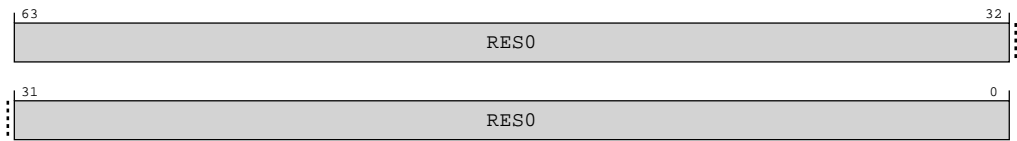


Table A-178: ID\_AA64DFR1\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

Access

MRS <Xt>, ID\_AA64DFR1\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64DFR1_EL1	0b11	0b000	0b0000	0b0101	0b001

Accessibility

MRS <Xt>, ID\_AA64DFR1\_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64DFR1_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64DFR1_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64DFR1_EL1;
```

## A.4.9 ID\_AA64AFR0\_EL1, AArch64 Auxiliary Feature Register 0

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'.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

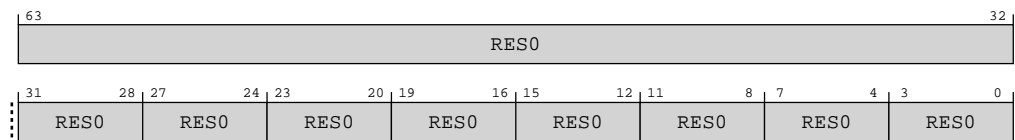
Identification

#### Reset value

0x0

### Bit descriptions

**Figure A-61: AArch64\_id\_aa64afr0\_el1 bit assignments**



**Table A-180: ID\_AA64AFR0\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0000

### Access

MRS <Xt>, ID\_AA64AFR0\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64AFR0_EL1	0b11	0b000	0b0000	0b0101	0b100

### Accessibility

MRS <Xt>, ID\_AA64AFR0\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then

```

```
if EL2Enabled() && HCR_EL2.TID3 == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
else
    return ID_AA64AFR0_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64AFR0_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64AFR0_EL1;
```

A.4.10 ID\_AA64AFR1\_EL1, AArch64 Auxiliary Feature Register 1

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'.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

0x0

Bit descriptions

Figure A-62: AArch64\_id\_aa64afr1\_el1 bit assignments

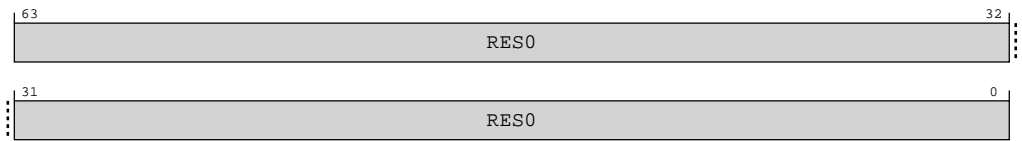


Table A-182: ID\_AA64AFR1\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

Access

MRS <Xt>, ID\_AA64AFR1\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64AFR1_EL1	0b11	0b000	0b0000	0b0101	0b101



## Accessibility

MRS <Xt>, ID\_AA64AFR1\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64AFR1_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64AFR1_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64AFR1_EL1;

```

### A.4.11 ID\_AA64ISAR0\_EL1, AArch64 Instruction Set Attribute Register 0

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'.

## Configurations

This register is available in all configurations.

## Attributes

### Width

64

### Functional group

Identification

### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-63: AArch64\_id\_aa64isar0\_el1 bit assignments**

63	60	59	56	55	52	51	48	47	44	43	40	39	36	35	32
RES0		TLB		TS		FHM		DP		SM4		SM3		SHA3	
31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0
	RDM		RES0		Atomic		CRC32		SHA2		SHA1		AES		RES0

**Table A-184: ID\_AA64ISAR0\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:60]	RES0	Reserved	0b0000
[59:56]	TLB	Indicates support for Outer Shareable and TLB range maintenance instructions. Defined values are: <b>0b0010</b> Outer Shareable and TLB range maintenance instructions are implemented.	
[55:52]	TS	Indicates support for flag manipulation instructions. Defined values are: <b>0b0010</b> CFINV, RMIF, SETF16, SETF8, AXFLAG, and XAFLAG instructions are implemented.	
[51:48]	FHM	Indicates support for FMLAL and FMLSL instructions. Defined values are: <b>0b0001</b> FMLAL and FMLSL instructions are implemented.	
[47:44]	DP	Indicates support for Dot Product instructions in AArch64 state. Defined values are: <b>0b0001</b> UDOT and SDOT instructions implemented.	
[43:40]	SM4	Indicates support for SM4 instructions in AArch64 state. Defined values are: <b>0b0000</b> When Cryptographic extensions are not implemented or disabled then SM3 instructions are not implemented. <b>0b0001</b> When Cryptographic extensions are implemented and enabled then SM3 instructions SM4E and SM4EKEY are implemented.	
[39:36]	SM3	Indicates support for SM3 instructions in AArch64 state. Defined values are: <b>0b0000</b> When Cryptographic extensions are not implemented or disabled then SM4 instructions are not implemented. <b>0b0001</b> When Cryptographic extensions are implemented and enabled then SM4 instructions SM3SS1, SM3TT1A, SM3TT1B, SM3TT2A, SM3TT2B, SM3PARTW1, and SM3PARTW2 are implemented.	
[35:32]	SHA3	Indicates support for SHA3 instructions in AArch64 state. Defined values are: <b>0b0000</b> When Cryptographic extensions are not implemented or disabled then SHA3 instructions are not implemented. <b>0b0001</b> When Cryptographic extensions are implemented and enabled then SHA3 instructions EOR3, RAX1, XAR, and BCAX are implemented.	
[31:28]	RDM	Indicates support for SQRDMLAH and SQRDMLSH instructions in AArch64 state. Defined values are: <b>0b0001</b> SQRDMLAH and SQRDMLSH instructions implemented.	
[27:24]	RES0	Reserved	0b0000
[23:20]	Atomic	Indicates support for Atomic instructions in AArch64 state. Defined values are: <b>0b0010</b> LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions implemented.	

Bits	Name	Description	Reset
[19:16]	CRC32	CRC32 instructions implemented in AArch64 state. Defined values are:  <b>0b0001</b> CRC32B, CRC32H, CRC32W, CRC32X, CRC32CB, CRC32CH, CRC32CW, and CRC32CX instructions implemented.	
[15:12]	SHA2	SHA2 instructions implemented in AArch64 state. Defined values are:  <b>0b0000</b> When Cryptographic extensions are not implemented or disabled then SHA2 instructions are not implemented.  <b>0b0010</b> When Cryptographic extensions are implemented and enabled then SHA256H, SHA256H2, SHA256SU0, SHA256SU1, SHA512H, SHA512H2, SHA512SU0, and SHA512SU1 instructions are implemented.  When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	
[11:8]	SHA1	SHA1 instructions implemented in AArch64 state. Defined values are:  <b>0b0000</b> When Cryptographic extensions are not implemented or disabled then SHA1 instructions are not implemented.  <b>0b0001</b> When Cryptographic extensions are implemented and enabled then SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 instructions are implemented.  When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	
[7:4]	AES	AES instructions implemented in AArch64 state. Defined values are:  <b>0b0000</b> When Cryptographic extensions are not implemented or disabled then AES instructions are not implemented.  <b>0b0010</b> When Cryptographic extensions are implemented and enabled then AESE, AESD, AESMC, and AESIMC instructions are implemented and also PMULL/PMULL2 instructions operating on 64-bit data quantities.  When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	
[3:0]	RES0	Reserved	0b0000

## Access

MRS <Xt>, ID\_AA64ISAR0\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64ISAR0_EL1	0b11	0b000	0b0000	0b0110	0b000

## Accessibility

MRS <Xt>, ID\_AA64ISAR0\_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64ISAR0_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64ISAR0_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64ISAR0_EL1;
```

### A.4.12 ID\_AA64ISAR1\_EL1, AArch64 Instruction Set Attribute Register 1

Provides information about the features and instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

#### Configurations

If ID\_AA64ISAR1\_EL1.{API, APA} == {0000, 0000}, then:

- The AArch64-TCR\_EL1.{TBID,TBID0}, AArch64-TCR\_EL2.{TBID0,TBID1}, AArch64-TCR\_EL2.TBID and AArch64-TCR\_EL3.TBID bits are RES0.
- AArch64-APIAKeyHi\_EL1, AArch64-APIAKeyLo\_EL1, AArch64-APIBKeyHi\_EL1, AArch64-APIBKeyLo\_EL1, AArch64-APDAKeyHi\_EL1, AArch64-APDAKeyLo\_EL1, AArch64-APDBKeyHi\_EL1, AArch64-APDBKeyLo\_EL1 are not allocated.
- 'SCTLR\_EL'.EnIA, 'SCTLR\_EL'.EnIB, 'SCTLR\_EL'.EnDA, 'SCTLR\_EL'.EnDB are all RES0.

If ID\_AA64ISAR1\_EL1.{GPI, GPA, API, APA} == {0000, 0000, 0000, 0000}, then:

- AArch64-HCR\_EL2.APK and AArch64-HCR\_EL2.API are RES0.
- AArch64-SCR\_EL3.APK and AArch64-SCR\_EL3.API are RES0.

#### Attributes

##### Width

64

##### Functional group

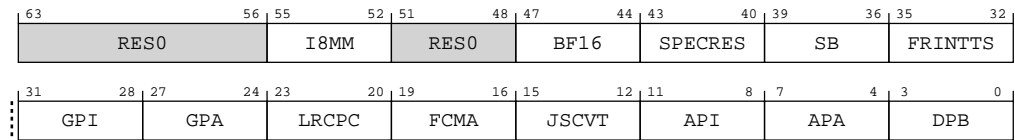
Identification

##### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-64: AArch64\_id\_aa64isar1\_el1 bit assignments**



**Table A-186: ID\_AA64ISAR1\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	0b00000000
[55:52]	I8MM	Indicates support for Advanced SIMD and Floating-point Int8 matrix multiplication instructions in AArch64 state. Defined values of this field are:  <b>0b0001</b> SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.	
[51:48]	RES0	Reserved	0b0000
[47:44]	BF16	Indicates support for Advanced SIMD and Floating-point BFloat16 instructions in AArch64 state. Defined values are:  <b>0b0001</b> BFDOT, BFMLAL, BFMLAL2, BFMMMLA, BFCVT, and BFCVT2 instructions are implemented.	
[43:40]	SPECRES	Indicates support for prediction invalidation instructions in AArch64 state. Defined values are:  <b>0b0001</b> CFP RCTX, DVP RCTX, and CPP RCTX instructions are implemented.	
[39:36]	SB	Indicates support for SB instruction in AArch64 state. Defined values are:  <b>0b0001</b> SB instruction is implemented.	
[35:32]	FRINTTS	Indicates support for the FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented. Defined values are:  <b>0b0001</b> FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented.	
[31:28]	GPI	Indicates support for an <b>IMPLEMENTATION DEFINED</b> algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are:  <b>0b0000</b> Generic Authentication using an IMPLEMENTATION DEFINED algorithm is not implemented.	
[27:24]	GPA	Indicates whether QARMA or Architected algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are:  <b>0b0001</b> Generic Authentication using the QARMA algorithm is implemented. This includes the PACGA instruction.	
[23:20]	LRCPC	Indicates support for weaker release consistency, RCpc, based model. Defined values are:  <b>0b0010</b> The LDAPR*, LDAPUR*, and STLUR* instructions are implemented.	

Bits	Name	Description	Reset
[19:16]	FCMA	Indicates support for complex number addition and multiplication, where numbers are stored in vectors. Defined values are:  <b>0b0001</b> The FCMLA and FCADD instructions are implemented.	
[15:12]	JSCVT	Indicates support for JavaScript conversion from double precision floating point values to integers in AArch64 state. Defined values are:  <b>0b0001</b> The FJCVTZS instruction is implemented.	
[11:8]	API	Indicates whether an <b>IMPLEMENTATION DEFINED</b> algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are:  <b>0b0000</b> Address Authentication using an IMPLEMENTATION DEFINED algorithm is not implemented.	
[7:4]	APA	Indicates whether QARMA or Architected algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are:  <b>0b0101</b> Address Authentication using the QARMA algorithm is implemented, with the HaveEnhancedPAC2() function returning TRUE, the HaveFPAC() function returning TRUE, the HaveFPACCombined() function returning TRUE, and the HaveEnhancedPAC() function returning FALSE.	
[3:0]	DPB	Data Persistence writeback. Indicates support for the rDC CVAP and rDC CVADP instructions in AArch64 state. Defined values are:  <b>0b0010</b> DC CVAP and DC CVADP supported	

## Access

MRS <Xt>, ID\_AA64ISAR1\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64ISAR1_EL1	0b11	0b000	0b0000	0b0110	0b001

## Accessibility

MRS <Xt>, ID\_AA64ISAR1\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64ISAR1_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64ISAR1_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64ISAR1_EL1;

```

### A.4.13 ID\_AA64MMFR0\_EL1, AArch64 Memory Model Feature Register 0

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'.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

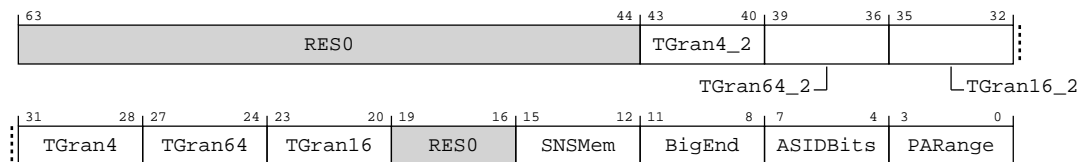
Identification

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure A-65: AArch64\_id\_aa64mmfr0\_el1 bit assignments**



**Table A-188: ID\_AA64MMFR0\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:44]	RES0	Reserved	0x0
[43:40]	TGran4_2	Indicates support for 4KB memory granule size for stage 2. Defined values are: <b>0b0010</b> 4KB granule supported at stage 2.	
[39:36]	TGran64_2	Indicates support for 64KB memory granule size for stage 2. Defined values are: <b>0b0010</b> 64KB granule supported at stage 2.	
[35:32]	TGran16_2	Indicates support for 16KB memory granule size for stage 2. Defined values are: <b>0b0010</b> 16KB granule supported at stage 2	

Bits	Name	Description	Reset
[31:28]	TGran4	Indicates support for 4KB memory translation granule size. Defined values are:  <b>0b0000</b> 4KB granule supported.	
[27:24]	TGran64	Indicates support for 64KB memory translation granule size. Defined values are:  <b>0b0000</b> 64KB granule supported.	
[23:20]	TGran16	Indicates support for 16KB memory translation granule size. Defined values are:  <b>0b0001</b> 16KB granule supported.	
[19:16]	RES0	Reserved	0b0000
[15:12]	SNSMem	Indicates support for a distinction between Secure and Non-secure Memory. Defined values are:  <b>0b0001</b> Does support a distinction between Secure and Non-secure Memory.	
[11:8]	BigEnd	Indicates support for mixed-endian configuration. Defined values are:  <b>0b0001</b> Mixed-endian support. The SCTLR_ELx.EE and SCTLR_EL1.EOE bits can be configured.	
[7:4]	ASIDBits	Number of ASID bits. Defined values are:  <b>0b0010</b> 16 bits.	
[3:0]	PARange	Physical Address range supported. Defined values are:  <b>0b0010</b> 40 bits, 1TB.	

## Access

MRS <Xt>, ID\_AA64MMFR0\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64MMFR0_EL1	0b11	0b000	0b0000	0b0111	0b000

## Accessibility

MRS <Xt>, ID\_AA64MMFR0\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64MMFR0_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64MMFR0_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64MMFR0_EL1;

```



## A.4.14 ID\_AA64MMFR1\_EL1, AArch64 Memory Model Feature Register 1

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'.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

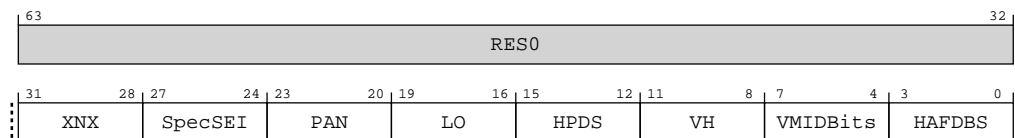
Identification

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-66: AArch64\_id\_aa64mmfr1\_el1 bit assignments**



**Table A-190: ID\_AA64MMFR1\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:28]	XNX	Indicates support for execute-never control distinction by Exception level at stage 2. Defined values are: <b>0b0001</b> Distinction between EL0 and EL1 execute-never control at stage 2 supported.	
[27:24]	SpecSEI	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: <b>0b0000</b> The PE never generates an SError interrupt due to an External abort on a speculative read.	
[23:20]	PAN	Privileged Access Never. Indicates support for the PAN bit in PSTATE, AArch64-SPSR_EL1, AArch64-SPSR_EL2, AArch64-SPSR_EL3, and AArch64-DSPSR_EL0. Defined values are: <b>0b0010</b> PAN supported and rAT S1E1RP and rAT S1E1WP instructions supported.	

Bits	Name	Description	Reset
[19:16]	LO	LORegions. Indicates support for LORegions. Defined values are:  <b>0b0001</b> LORegions supported.	
[15:12]	HPDS	Hierarchical Permission Disables. Indicates support for disabling hierarchical controls in translation tables. Defined values are:  <b>0b0010</b> Disabling of hierarchical controls supported with the TCR_EL1.{HPD1, HPD0}, TCR_EL2.HPD or TCR_EL2.{HPD1, HPD0}, and TCR_EL3.HPD bits and adds possible hardware allocation of bits[62:59] of the translation table descriptors from the final lookup level for IMPLEMENTATION DEFINED use.	
[11:8]	VH	Virtualization Host Extensions. Defined values are:  <b>0b0001</b> Virtualization Host Extensions supported.	
[7:4]	VMIDBits	Number of VMID bits. Defined values are:  <b>0b0010</b> 16 bits	
[3:0]	HAFDBS	Hardware updates to Access flag and Dirty state in translation tables. Defined values are:  <b>0b0010</b> Hardware update of both the Access flag and dirty state is supported.	

## Access

MRS <Xt>, ID\_AA64MMFR1\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64MMFR1_EL1	0b11	0b000	0b0000	0b0111	0b001

## Accessibility

MRS <Xt>, ID\_AA64MMFR1\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64MMFR1_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64MMFR1_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64MMFR1_EL1;

```

## A.4.15 ID\_AA64MMFR2\_EL1, AArch64 Memory Model Feature Register 2

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'.

### Configurations



Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

### Attributes

#### Width

64

#### Functional group

Identification

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-67: AArch64\_id\_aa64mmfr2\_el1 bit assignments**

63	60	59	56	55	52	51	48	47	44	43	40	39	36	35	32
EOPD	EVT	BBM	TTL	RES0	FWB	IDS	AT								
31	28	27	24	23	20	19	16	15	12	11	8	7	4	3	0
ST	RES0	CCIDX	RES0	IESB	RES0	UAO	CnP								

**Table A-192: ID\_AA64MMFR2\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:60]	EOPD	Indicates support for the EOPD mechanism. Defined values are: <b>0b0001</b> EOPDx mechanism is implemented.	
[59:56]	EVT	Enhanced Virtualization Traps. If EL2 is implemented, indicates support for the AArch64-HCR_EL2.{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps. Defined values are: <b>0b0010</b> AArch64-HCR_EL2.{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps are supported.	

Bits	Name	Description	Reset
[55:52]	BBM	Allows Identification of the requirements of the hardware to have break-before-make sequences when changing block size for a translation. <b>0b0010</b> Level 2 support for changing block size is supported.	
[51:48]	TTL	Indicates support for TTL field in address operations. Defined values are: <b>0b0001</b> TLB maintenance instructions by address have bits[47:44] holding the TTL field.	
[47:44]	RES0	Reserved	0b0000
[43:40]	FWB	Indicates support for AArch64-HCR_EL2.FWB. Defined values are: <b>0b0001</b> AArch64-HCR_EL2.FWB is supported.	
[39:36]	IDS	Indicates the value of ESR_ELx.EC that reports an exception generated by a read access to the feature ID space. Defined values are: <b>0b0001</b> All exceptions generated by an AArch64 read access to the feature ID space are reported by ESR_ELx.EC == 0x18.	
[35:32]	AT	Identifies support for unaligned single-copy atomicity and atomic functions. Defined values are: <b>0b0001</b> Unaligned single-copy atomicity and atomic functions with a 16-byte address range aligned to 16-bytes are supported.	
[31:28]	ST	Identifies support for small translation tables. Defined values are: <b>0b0001</b> The maximum value of the TCR_ELx.{TOSZ,T1SZ} and VTCR_EL2.TOSZ fields is 48 for 4KB and 16KB granules, and 47 for 64KB granules.	
[27:24]	RES0	Reserved	0b0000
[23:20]	CCIDX	Support for the use of revised AArch64-CCSIDR_EL1 register format. Defined values are: <b>0b0001</b> 64-bit format implemented for all levels of the CCSIDR_EL1.	
[19:16]	RES0	Reserved	0b0000
[15:12]	IESB	Indicates support for the IESB bit in the SCTLR_ELx registers. Defined values are: <b>0b0001</b> IESB bit in the SCTLR_ELx registers is supported.	
[11:8]	RES0	Reserved	0b0000
[7:4]	UAO	User Access Override. Defined values are: <b>0b0001</b> UAO supported.	
[3:0]	CnP	Indicates support for Common not Private translations. Defined values are: <b>0b0001</b> Common not Private translations supported.	

## Access

MRS <Xt>, ID\_AA64MMFR2\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64MMFR2_EL1	0b11	0b000	0b0000	0b0111	0b010

## Accessibility

MRS <Xt>, ID\_AA64MMFR2\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && (!IsZero(ID_AA64MMFR2_EL1) || boolean IMPLEMENTATION_DEFINED
        "ID_AA64MMFR2 trapped by HCR_EL2.TID3") && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64MMFR2_EL1;
elseif PSTATE.EL == EL2 then
    return ID_AA64MMFR2_EL1;
elseif PSTATE.EL == EL3 then
    return ID_AA64MMFR2_EL1;

```

## A.4.16 CLIDR\_EL1, Cache Level ID Register

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.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

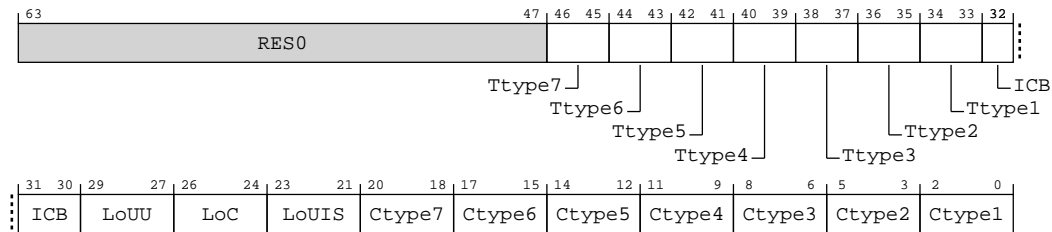
Identification

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-68: AArch64\_clidr\_el1 bit assignments**



**Table A-194: CLIDR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:47]	RES0	Reserved	0x0
[46:45]	Ttype7	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy.  <b>0b00</b> No Tag Cache.	
[44:43]	Ttype6	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy.  <b>0b00</b> No Tag Cache.	
[42:41]	Ttype5	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy.  <b>0b00</b> No Tag Cache.	
[40:39]	Ttype4	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy.  <b>0b00</b> No Tag Cache.	
[38:37]	Ttype3	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy.  <b>0b00</b> When no L3 present, no tag cache.  <b>0b10</b> When L3 present, Unified Allocation Tag and Data cache at L3	
[36:35]	Ttype2	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy.  <b>0b10</b> Unified Allocation Tag and Data cache at L1	

Bits	Name	Description	Reset
[34:33]	Ttype1	Tag cache type. Indicate the type of cache that is implemented and can be managed using the architected cache maintenance instructions that operate by set/way at each level, from Level 1 up to a maximum of seven levels of cache hierarchy.  <b>0b10</b> Unified Allocation Tag and Data cache at L1	
[32:30]	ICB	Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions.  The possible values are:  <b>0b000</b> Not disclosed by this mechanism.  <b>0b001</b> L1 cache is the highest Inner Cacheable level.  <b>0b010</b> L2 cache is the highest Inner Cacheable level.  <b>0b011</b> L3 cache is the highest Inner Cacheable level.  <b>0b100</b> L4 cache is the highest Inner Cacheable level.  <b>0b101</b> L5 cache is the highest Inner Cacheable level.  <b>0b110</b> L6 cache is the highest Inner Cacheable level.  <b>0b111</b> L7 cache is the highest Inner Cacheable level.	
[29:27]	LoUU	Level of Unification Uniprocessor for the cache hierarchy.  <b>0b000</b> Level of Unification Uniprocessor is before the L1 D-cache.	
[26:24]	LoC	Level of Coherence for the cache hierarchy.  <b>0b010</b> When no L3 present, Level 2  <b>0b011</b> When L3 present, Level 3	
[23:21]	LoUIS	Level of Unification Inner Shareable for the cache hierarchy.  <b>0b000</b> No cache level needs cleaning to Point of Unification	
[20:18]	Ctype7	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:  <b>0b000</b> No cache.	

Bits	Name	Description	Reset
[17:15]	Ctype6	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:  <b>0b000</b> No cache.	
[14:12]	Ctype5	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:  <b>0b000</b> No cache.	
[11:9]	Ctype4	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:  <b>0b000</b> No cache.	
[8:6]	Ctype3	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:  <b>0b000</b> No L3.  <b>0b100</b> Unified instruction and data caches at L3	
[5:3]	Ctype2	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:  <b>0b100</b> Unified instruction and data caches at L2	
[2:0]	Ctype1	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:  <b>0b011</b> Separate instruction and data caches at L1	

## Access

MRS <Xt>, CLIDR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
CLIDR_EL1	0b11	0b001	0b0000	0b0000	0b001

## Accessibility

MRS <Xt>, CLIDR\_EL1

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);

```



```
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.TID4 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.CLIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CLIDR_EL1;
elseif PSTATE.EL == EL2 then
    return CLIDR_EL1;
elseif PSTATE.EL == EL3 then
    return CLIDR_EL1;
```

A.4.17 GMID\_EL1, Multiple tag transfer ID register

Indicates the block size that is accessed by the LDGM and STGM System instructions.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure A-69: AArch64\_gmid\_el1 bit assignments

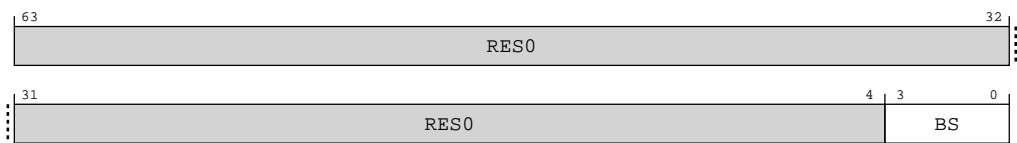


Table A-196: GMID\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	0x0
[3:0]	BS	Log <sub>2</sub> of the block size in words. The minimum supported size is 16B (value == 2) and the maximum is 256B (value == 6).  0b0100 Log2 of the block size is 4	



**Table A-198: CTR\_EL0 bit descriptions**

Bits	Name	Description	Reset
[63:38]	RES0	Reserved	0x0
[37:32]	TminLine	<p>Tag minimum Line. <math>\log_2</math> of the number of words covered by Allocation Tags in the smallest cache line of all caches which can contain Allocation tags that are controlled by the PE.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>For an implementation with cache lines containing 64 bytes of data and 4 Allocation Tags, this will be <math>\log_2(64/4) = 4</math>.</li> <li>For an implementation with Allocations Tags in separate cache lines of 128 Allocation Tags per line, this will be <math>\log_2(128*16/4) = 9</math>.</li> </ul> <p><b>0b000100</b>  <math>\log_2</math> of number of words (<math>64/4=16</math>) covered by Allocation Tags in the smallest cache line of all caches</p>	
[31]	RES1	Reserved	0b1
[30]	RES0	Reserved	0b0
[29]	DIC	<p>Instruction cache invalidation requirements for data to instruction coherence.</p> <p><b>0b0</b>  Instruction cache invalidation to the point of unification is required for instruction to data coherence.</p>	
[28]	IDC	<p>Data cache clean requirements for instruction to data coherence. The meaning of this bit is:</p> <p><b>0b1</b>  Data cache clean to the Point of Unification is not required for instruction to data coherence.</p>	
[27:24]	CWG	<p>Cache writeback granule. <math>\log_2</math> 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.</p> <p><b>0b0100</b>  64 bytes.</p>	
[23:20]	ERG	<p>Exclusives reservation granule, and, if TME is implemented, transactional reservation granule. <math>\log_2</math> of the number of words of the maximum size of the reservation granule for the Load-Exclusive and Store-Exclusive instructions, and, if TME is implemented, for detecting transactional conflicts.</p> <p><b>0b0100</b>  64 bytes.</p>	
[19:16]	DminLine	<p><math>\log_2</math> of the number of words in the smallest cache line of all the data caches and unified caches that are controlled by the PE.</p> <p><b>0b0100</b>  64 bytes.</p>	
[15:14]	L1lp	<p>Level 1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache. Possible values of this field are:</p> <p><b>0b11</b>  Physical Index, Physical Tag (PIPT)</p>	
[13:4]	RES0	Reserved	0x0
[3:0]	IminLine	<p><math>\log_2</math> of the number of words in the smallest cache line of all the instruction caches that are controlled by the PE.</p> <p><b>0b0100</b>  64 bytes.</p>	

## Access

MRS <Xt>, CTR\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
CTR_ELO	0b11	0b011	0b0000	0b0000	0b001

## Accessibility

MRS <Xt>, CTR\_ELO

```

if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11') && SCTLR_EL1.UCT == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' && HF\
GRTR_EL2.CTR_ELO == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11' && SCTLR_EL2.UCT == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CTR_ELO;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.CTR_ELO == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CTR_ELO;
elseif PSTATE.EL == EL2 then
    return CTR_ELO;
elseif PSTATE.EL == EL3 then
    return CTR_ELO;

```

## A.4.19 DCZID\_ELO, Data Cache Zero ID register

Indicates the block size that is written with byte values of 0 by the rDC ZVA (Data Cache Zero by Address) System instruction.

If ARMv8.5-MemTag is implemented, this register also indicates the granularity at which the rDC GVA and rDC GZVA instructions write.

## Configurations

This register is available in all configurations.

## Attributes

### Width

64

### Functional group

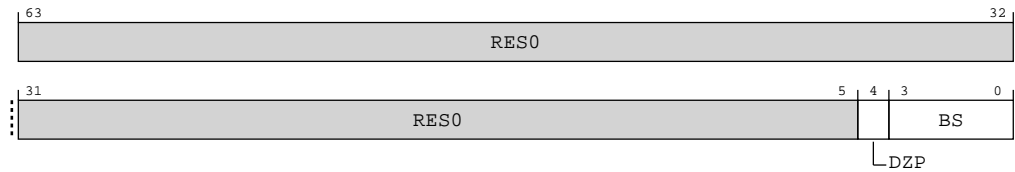
Identification

## Reset value

See individual bit resets.

## Bit descriptions

**Figure A-71: AArch64\_dczip\_el0 bit assignments**



**Table A-200: DCZID\_EL0 bit descriptions**

Bits	Name	Description	Reset
[63:5]	RES0	Reserved	0x0
[4]	DZP	Data Zero Prohibited. This field indicates whether use of rDC ZVA instructions is permitted or prohibited.  If ARMv8.5-MemTag is implemented, this field also indicates whether use of the rDC GVA and rDC GZVA instructions are permitted or prohibited.  <b>0b0</b> Instructions are permitted.	
[3:0]	BS	Log <sub>2</sub> of the block size in words. The maximum size supported is 2KB (value == 9).  <b>0b0100</b> Log2 of the block size is 4	

## Access

MRS <Xt>, DCZID\_EL0

<systemreg>	op0	op1	CRn	CRm	op2
DCZID_EL0	0b11	0b011	0b0000	0b0000	0b111

## Accessibility

MRS <Xt>, DCZID\_EL0

```

if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' && HF\
GRTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return DCZID_EL0;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGRTR_EL2.DCZID_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return DCZID_EL0;
elseif PSTATE.EL == EL2 then
    return DCZID_EL0;
elseif PSTATE.EL == EL3 then
    return DCZID_EL0;

```

## A.4.20 MPAMIDR\_EL1, MPAM ID Register (EL1)

Indicates the presence and maximum PARTID and PMG values supported in the implementation. It also indicates whether the implementation supports MPAM virtualization.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

Identification

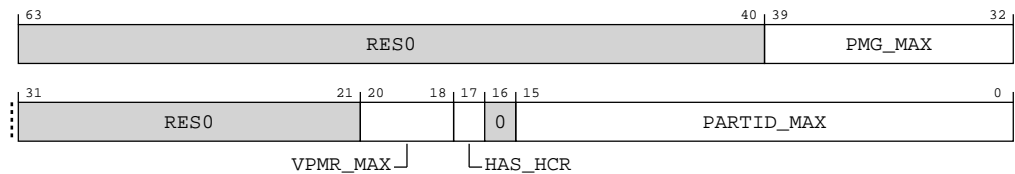
#### Reset value

See individual bit resets.

### Bit descriptions

MPAMIDR\_EL1 indicates the MPAM implementation parameters of the PE.

**Figure A-72: AArch64\_mpamidr\_el1 bit assignments**



**Table A-202: MPAMIDR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	0x0
[39:32]	PMG_MAX	The largest value of PMG that the implementation can generate. The PMG_I and PMG_D fields of every MPAMn_ELx must implement at least enough bits to represent PMG_MAX. <b>0b00000001</b> Max PMG field is 1 (1-bit)	
[31:21]	RES0	Reserved	0x0
[20:18]	VPMR_MAX	If HAS_HCR == 0, VPMR_MAX must be 0b000. Otherwise, it indicates the maximum register index n for the MPAMVPM<n>_EL2 registers. <b>0b001</b> 2 MPAMVPMn_EL2 registers are implemented	

Bits	Name	Description	Reset
[17]	HAS_HCR	HAS_HCR indicates that the PE implementation supports MPAM virtualization, including AArch64-MPAMHCR_EL2, AArch64-MPAMVPMV_EL2 and MPAMVPM<n>_EL2 with n in the range 0 to VPMR_MAX. Must be 0 if EL2 is not implemented in either security state.  <b>0b1</b> MPAM virtualization is supported.	
[16]	RESO	Reserved	0b0
[15:0]	PARTID_MAX	The largest value of PARTID that the implementation can generate. The PARTID_I and PARTID_D fields of every MPAMn_ELx must implement at least enough bits to represent PARTID_MAX.  <b>0b0000000000011111</b> Max PARTID field is 63	

### Access

MRS <Xt>, MPAMIDR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MPAMIDR_EL1	0b11	0b000	0b1010	0b0100	0b100

### Accessibility

MRS <Xt>, MPAMIDR\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && MPAMHCR_EL2.TRAP_MPAMIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return MPAMIDR_EL1;
elsif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return MPAMIDR_EL1;
elsif PSTATE.EL == EL3 then
    return MPAMIDR_EL1;

```

## A.4.21 IMP\_CPUCFR\_EL1, CPU Configuration Register

This register provides configuration information for the core.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure A-73: AArch64\_imp\_cpucfr\_el1 bit assignments

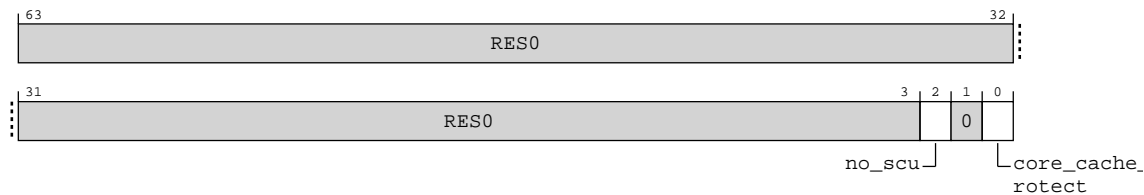


Table A-204: IMP\_CPUCFR\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	0x0
[2]	no_scu	Indicates whether the SCU is present or not. Possible values of this bit are:  <b>0b0</b> The SCU is present.  <b>0b1</b> The SCU is not present.	
[1]	RES0	Reserved	0x0
[0]	core_cache_protect	Indicates whether ECC is present or not. Possible values of this field are:  <b>0b0</b> ECC is not present.  <b>0b1</b> ECC is present.	

Access

MRS <Xt>, S3\_O\_C15\_CO\_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_O_C15_CO_0	0b11	0b000	0b1111	0b0000	0b000

Accessibility

MRS <Xt>, S3\_O\_C15\_CO\_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return IMP_CPUCFR_EL1;
```



```

elseif PSTATE.EL == EL2 then
    return IMP_CPUCFR_EL1;
elseif PSTATE.EL == EL3 then
    return IMP_CPUCFR_EL1;

```

## A.5 AArch64 Performance Monitors register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Performance Monitors registers in the core. Individual register descriptions provide detailed information.

**Table A-206: Performance Monitors register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
PMMIR_EL1	3	C9	0	C14	6	See individual bit resets.	64-bit	Performance Monitors Machine Identification Register
PMCR_ELO	3	C9	3	C12	0	See individual bit resets.	64-bit	Performance Monitors Control Register
PMCEID0_ELO	3	C9	3	C12	6	See individual bit resets.	64-bit	Performance Monitors Common Event Identification register 0
PMCEID1_ELO	3	C9	3	C12	7	See individual bit resets.	64-bit	Performance Monitors Common Event Identification register 1

### A.5.1 PMMIR\_EL1, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation to software.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

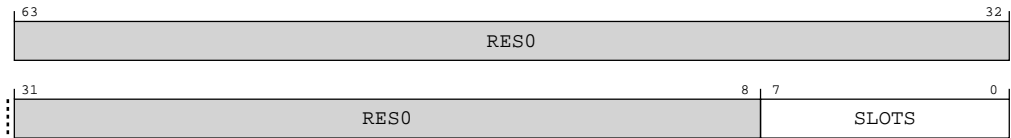
Performance Monitors

##### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-74: AArch64\_pmmir\_el1 bit assignments**



**Table A-207: PMMIR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:8]	RES0	Reserved	0x0
[7:0]	SLOTS	Operation width. The largest value by which the STALL_SLOT event might increment by in a single cycle. If the STALL_SLOT event is not implemented, this field might read as zero.  <b>0b00001000</b> The largest value by which the STALL_SLOT PMU event may increment in one cycle is 8.	

## Access

MRS <Xt>, PMMIR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
PMMIR_EL1	0b11	0b000	0b1001	0b1110	0b110

## Accessibility

MRS <Xt>, PMMIR\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.PMMIR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMMIR_EL1;
elseif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMMIR_EL1;
elseif PSTATE.EL == EL3 then
    return PMMIR_EL1;

```

## A.5.2 PMCR\_EL0, Performance Monitors Control Register

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

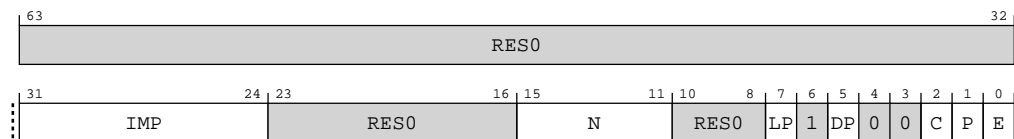
Performance Monitors

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-75: AArch64\_pmcr\_el0 bit assignments**



**Table A-209: PMCR\_EL0 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:24]	IMP	Implementer code: <b>0b00000000</b> No ID information is present in PMCR/PMCR_EL0. Software must use the MIDR_EL1 to identify the PE.	
[23:16]	RES0	Reserved	0b00000000
[15:11]	N	Number of event counters: <b>0b00110</b> When configured for 6 PMU counters, denotes 6 PMU counters implemented <b>0b10100</b> When configured for 20 PMU counters, denotes 20 PMU counters implemented	
[10:8]	RES0	Reserved	0b000

Bits	Name	Description	Reset
[7]	LP	<p>Long event counter enable. Determines when unsigned overflow is recorded by a counter overflow bit.</p> <p><b>0b0</b></p> <p>Event counter overflow on increment that causes unsigned overflow of AArch64-PMEVCNTR&lt;n&gt;_ELO[31:0].</p> <p><b>0b1</b></p> <p>Event counter overflow on increment that causes unsigned overflow of AArch64-PMEVCNTR&lt;n&gt;_ELO[63:0].</p> <p>If EL2 is implemented and AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN is less than PMCR_ELO.N, this bit does not affect the operation of event counters in the range [AArch32-HDCR.HPMN..(PMCR_ELO.N-1)] or [AArch64-MDCR_EL2.HPMN..(PMCR_ELO.N-1)].</p> <p><b>Note:</b></p> <p>The effect of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state. For more information, see the description of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN.</p>	
[6]	RES1	Reserved	0b1
[5]	DP	<p>Disable cycle counter when event counting is prohibited.</p> <p><b>0b0</b></p> <p>Cycle counting by AArch64-PMCCNTR_ELO is not affected by this bit.</p> <p><b>0b1</b></p> <p>When event counting for counters in the range [0..(AArch64-MDCR_EL2.HPMN-1)] is prohibited, cycle counting by AArch64-PMCCNTR_ELO is disabled.</p>	
[4:3]	RES0	Reserved	0b0
[2]	C	<p>Cycle counter reset. The effects of writing to this bit are:</p> <p><b>0b0</b></p> <p>No action.</p> <p><b>0b1</b></p> <p>Reset AArch64-PMCCNTR_ELO to zero.</p> <p>This bit is always RAZ.</p> <p><b>Note:</b></p> <p>Resetting AArch64-PMCCNTR_ELO does not change the cycle counter overflow bit. The value of PMCR_ELO.LC is ignored, and bits [63:0] of all affected event counters are reset.</p>	

Bits	Name	Description	Reset
[1]	P	<p>Event counter reset. The effects of writing to this bit are:</p> <p><b>0b0</b></p> <p>No action.</p> <p><b>0b1</b></p> <p>Reset all event counters accessible in the current Exception level, not including AArch64-PMCCNTR_ELO, to zero.</p> <p>This bit is always RAZ.</p> <p>In EL0 and EL1:</p> <ul style="list-style-type: none"> <li>If EL2 is implemented and enabled in the current Security state, and AArch64-MDCR_EL2.HPMN is less than PMCR_ELO.N, a write of 1 to this bit does not reset event counters in the range [AArch64-MDCR_EL2.HPMN..(PMCR_ELO.N-1)].</li> <li>If EL2 is not implemented, EL2 is disabled in the current Security state, or AArch64-MDCR_EL2.HPMN equals PMCR_ELO.N, a write of 1 to this bit resets all the event counters.</li> </ul> <p>In EL2 and EL3, a write of 1 to this bit resets all the event counters.</p> <p><b>Note:</b> Resetting the event counters does not change the event counter overflow bits.</p> <p>If ARMv8.5-PMU is implemented, the values of AArch64-MDCR_EL2.HLP and PMCR_ELO.LP are ignored, and bits [63:0] of all affected event counters are reset.</p>	
[0]	E	<p>Enable.</p> <p><b>0b0</b></p> <p>All event counters in the range [0..(PMN-1)] and AArch64-PMCCNTR_ELO, are disabled.</p> <p><b>0b1</b></p> <p>All event counters in the range [0..(PMN-1)] and AArch64-PMCCNTR_ELO, are enabled by AArch64-PMCNTENSET_ELO.</p> <p>If EL2 is implemented, then:</p> <ul style="list-style-type: none"> <li>If EL2 is using AArch64, PMN is AArch64-MDCR_EL2.HPMN.</li> <li>If PMN is less than PMCR_ELO.N, this bit does not affect the operation of event counters in the range [PMN..(PMCR_ELO.N-1)].</li> </ul> <p>If EL2 is not implemented, PMN is PMCR_ELO.N.</p> <p><b>Note:</b> The effect of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN on the operation of this bit always applies if EL2 is implemented, at all Exception levels including EL2 and EL3, and regardless of whether EL2 is enabled in the current Security state. For more information, see the description of AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN.</p>	

## Access

MRS <Xt>, PMCR\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
PMCR_ELO	0b11	0b011	0b1001	0b1100	0b000

MSR PMCR\_ELO, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
PMCR_ELO	0b11	0b011	0b1001	0b1100	0b000

## Accessibility

MRS &lt;Xt&gt;, PMCR\_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCR_ELO;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCR_ELO;
    elsif PSTATE.EL == EL2 then
        if MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCR_ELO;
    elsif PSTATE.EL == EL3 then
        return PMCR_ELO;

```

MSR PMCR\_ELO, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11' && SCR_EL3.FGTEn == '1' && HD\
FGWTR_EL2.PMCR_ELO == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCR_ELO = X[t];
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && SCR_EL3.FGTEn == '1' && HDBGWTR_EL2.PMCR_ELO == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);

```

```

    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif MDCR_EL3.TPM == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t];
    elsif PSTATE.EL == EL2 then
        if MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCR_EL0 = X[t];
    elsif PSTATE.EL == EL3 then
        PMCR_EL0 = X[t];

```

### A.5.3 PMCEID0\_ELO, Performance Monitors Common Event Identification register 0

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0000 to 0x001F and 0x4000 to 0x401F.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. For more information about the common events and the use of the PMCEID<n>\_ELO registers see 'The PMU event number space and common events'.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

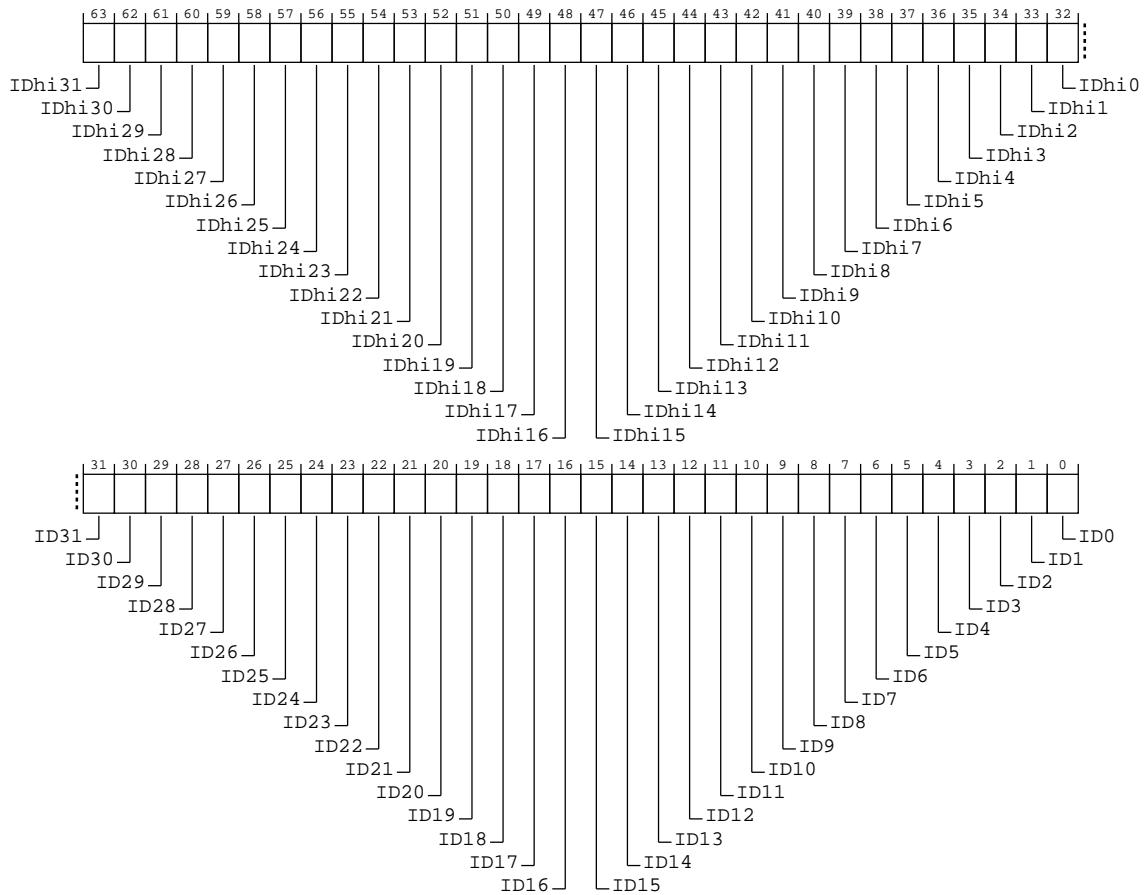
Performance Monitors

##### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-76: AArch64\_pmceid0\_el0 bit assignments**



**Table A-212: PMCEID0\_ELO bit descriptions**

Bits	Name	Description	Reset
[63]	IDHi31	IDHi31 corresponds to a Reserved Event event (0x401f) <b>0b0</b> The common event is not implemented, or not counted.	
[62]	IDHi30	IDHi30 corresponds to a Reserved Event event (0x401e) <b>0b0</b> The common event is not implemented, or not counted.	
[61]	IDHi29	IDHi29 corresponds to a Reserved Event event (0x401d) <b>0b0</b> The common event is not implemented, or not counted.	
[60]	IDHi28	IDHi28 corresponds to a Reserved Event event (0x401c) <b>0b0</b> The common event is not implemented, or not counted.	



Bits	Name	Description	Reset
[59]	IDhi27	IDhi27 corresponds to common event (0x401b) CTI_TRIGOUT7 <b>0b1</b> The common event is implemented.	
[58]	IDhi26	IDhi26 corresponds to common event (0x401a) CTI_TRIGOUT6 <b>0b1</b> The common event is implemented.	
[57]	IDhi25	IDhi25 corresponds to common event (0x4019) CTI_TRIGOUT5 <b>0b1</b> The common event is implemented.	
[56]	IDhi24	IDhi24 corresponds to common event (0x4018) CTI_TRIGOUT4 <b>0b1</b> The common event is implemented.	
[55]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4017) <b>0b0</b> The common event is not implemented, or not counted.	
[54]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4016) <b>0b0</b> The common event is not implemented, or not counted.	
[53]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4015) <b>0b0</b> The common event is not implemented, or not counted.	
[52]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4014) <b>0b0</b> The common event is not implemented, or not counted.	
[51]	IDhi19	IDhi19 corresponds to common event (0x4013) TRCEXTOUT3 <b>0b1</b> The common event is implemented.	
[50]	IDhi18	IDhi18 corresponds to common event (0x4012) TRCEXTOUT2 <b>0b1</b> The common event is implemented.	
[49]	IDhi17	IDhi17 corresponds to common event (0x4011) TRCEXTOUT1 <b>0b1</b> The common event is implemented.	
[48]	IDhi16	IDhi16 corresponds to common event (0x4010) TRCEXTOUT0 <b>0b1</b> The common event is implemented.	
[47]	IDhi15	IDhi15 corresponds to common event (0x400f) Reserved <b>0b0</b> The common event is not implemented, or not counted.	
[46]	IDhi14	IDhi14 corresponds to common event (0x400e) TRB_TRIG <b>0b0</b> The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[45]	IDHi13	IDHi13 corresponds to common event (0x400d) PMU_OVFS <b>0b0</b> The common event is not implemented, or not counted.	
[44]	IDHi12	IDHi12 corresponds to common event (0x400c) TRB_WRAP <b>0b1</b> The common event is implemented.	
[43]	IDHi11	IDHi11 corresponds to common event (0x400b) L3D_CACHE_LMISS_RD <b>0b1</b> The common event is implemented.	
[42]	IDHi10	IDHi10 corresponds to common event (0x400a) L2I_CACHE_LMISS <b>0b0</b> The common event is not implemented, or not counted.	
[41]	IDHi9	IDHi9 corresponds to common event (0x4009) L2D_CACHE_LMISS_RD <b>0b1</b> The common event is implemented.	
[40]	IDHi8	IDHi8 corresponds to common event (0x4008) Reserved <b>0b0</b> The common event is not implemented, or not counted.	
[39]	IDHi7	IDHi7 corresponds to common event (0x4007) Reserved <b>0b0</b> The common event is not implemented, or not counted.	
[38]	IDHi6	IDHi6 corresponds to common event (0x4006) L1I_CACHE_LMISS <b>0b1</b> The common event is implemented.	
[37]	IDHi5	IDHi5 corresponds to common event (0x4005) STALL_BACKEND_MEM <b>0b1</b> The common event is implemented.	
[36]	IDHi4	IDHi4 corresponds to common event (0x4004) CNT_CYCLES <b>0b1</b> The common event is implemented.	
[35]	IDHi3	IDHi3 corresponds to common event (0x4003) SAMPLE_COLLISION <b>0b0</b> The common event is not implemented, or not counted.	
[34]	IDHi2	IDHi2 corresponds to common event (0x4002) SAMPLE_FILTRATE <b>0b0</b> The common event is not implemented, or not counted.	
[33]	IDHi1	IDHi1 corresponds to common event (0x4001) SAMPLE_FEED <b>0b0</b> The common event is not implemented, or not counted.	
[32]	IDHi0	IDHi0 corresponds to common event (0x4000) SAMPLE_POP <b>0b0</b> The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x1f) L1D_CACHE_ALLOCATE <b>0b0</b> The common event is not implemented, or not counted.	
[30]	ID30	ID30 corresponds to common event (0x1e) CHAIN <b>0b1</b> The common event is implemented.	
[29]	ID29	ID29 corresponds to common event (0x1d) BUS_CYCLES <b>0b1</b> The common event is implemented.	
[28]	ID28	ID28 corresponds to common event (0x1c) TTBR_WRITE_RETIRED <b>0b1</b> The common event is implemented.	
[27]	ID27	ID27 corresponds to common event (0x1b) INST_SPEC <b>0b1</b> The common event is implemented.	
[26]	ID26	ID26 corresponds to common event (0x1a) MEMORY_ERROR <b>0b1</b> The common event is implemented.	
[25]	ID25	ID25 corresponds to common event (0x19) BUS_ACCESS <b>0b1</b> The common event is implemented.	
[24]	ID24	ID24 corresponds to common event (0x18) L2D_CACHE_WB <b>0b1</b> The common event is implemented.	
[23]	ID23	ID23 corresponds to common event (0x17) L2D_CACHE_REFILL <b>0b1</b> The common event is implemented.	
[22]	ID22	ID22 corresponds to common event (0x16) L2D_CACHE <b>0b1</b> The common event is implemented.	
[21]	ID21	ID21 corresponds to common event (0x15) L1D_CACHE_WB <b>0b1</b> The common event is implemented.	
[20]	ID20	ID20 corresponds to common event (0x14) L1I_CACHE <b>0b1</b> The common event is implemented.	
[19]	ID19	ID19 corresponds to common event (0x13) MEM_ACCESS <b>0b1</b> The common event is implemented.	
[18]	ID18	ID18 corresponds to common event (0x12) BR_PRED <b>0b1</b> The common event is implemented.	

Bits	Name	Description	Reset
[17]	ID17	ID17 corresponds to common event (0x11) CPU_CYCLES <b>0b1</b> The common event is implemented.	
[16]	ID16	ID16 corresponds to common event (0x10) BR_MIS_PRED <b>0b1</b> The common event is implemented.	
[15]	ID15	ID15 corresponds to common event (0xf) UNALIGNED_LDST_RETIRED <b>0b0</b> The common event is not implemented, or not counted.	
[14]	ID14	ID14 corresponds to common event (0xe) BR_RETURN_RETIRED <b>0b0</b> The common event is not implemented, or not counted.	
[13]	ID13	ID13 corresponds to common event (0xd) BR_IMMED_RETIRED <b>0b0</b> The common event is not implemented, or not counted.	
[12]	ID12	ID12 corresponds to common event (0xc) PC_WRITE_RETIRED <b>0b0</b> The common event is not implemented, or not counted.	
[11]	ID11	ID11 corresponds to common event (0xb) CID_WRITE_RETIRED <b>0b1</b> The common event is implemented.	
[10]	ID10	ID10 corresponds to common event (0xa) EXC_RETURN <b>0b1</b> The common event is implemented.	
[9]	ID9	ID9 corresponds to common event (0x9) EXC_TAKEN <b>0b1</b> The common event is implemented.	
[8]	ID8	ID8 corresponds to common event (0x8) INST_RETIRED <b>0b1</b> The common event is implemented.	
[7]	ID7	ID7 corresponds to common event (0x7) ST_RETIRED <b>0b0</b> The common event is not implemented, or not counted.	
[6]	ID6	ID6 corresponds to common event (0x6) LD_RETIRED <b>0b0</b> The common event is not implemented, or not counted.	
[5]	ID5	ID5 corresponds to common event (0x5) L1D_TLB_REFILL <b>0b1</b> The common event is implemented.	
[4]	ID4	ID4 corresponds to common event (0x4) L1D_CACHE <b>0b1</b> The common event is implemented.	

Bits	Name	Description	Reset
[3]	ID3	ID3 corresponds to common event (0x3) L1D_CACHE_REFILL  <b>0b1</b> The common event is implemented.	
[2]	ID2	ID2 corresponds to common event (0x2) L1I_TLB_REFILL  <b>0b1</b> The common event is implemented.	
[1]	ID1	ID1 corresponds to common event (0x1) L1I_CACHE_REFILL  <b>0b1</b> The common event is implemented.	
[0]	ID0	ID0 corresponds to common event (0x0) SW_INCR  <b>0b1</b> The common event is implemented.	

## Access

MRS <Xt>, PMCEID0\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
PMCEID0_ELO	0b11	0b011	0b1001	0b1100	0b110

## Accessibility

MRS <Xt>, PMCEID0\_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCEID0_ELO;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCEID0_ELO;
    elsif PSTATE.EL == EL2 then
        if MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCEID0_ELO;
    elsif PSTATE.EL == EL3 then
        return PMCEID0_ELO;

```

## A.5.4 PMCEID1\_ELO, Performance Monitors Common Event Identification register 1

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the ranges 0x0020 to 0x003F and 0x4020 to 0x403F.

When the value of a bit in the register is 1 the corresponding common event is implemented and counted. Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0. For more information about the common events and the use of the PMCEID<n>\_ELO registers see 'The PMU event number space and common events'.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

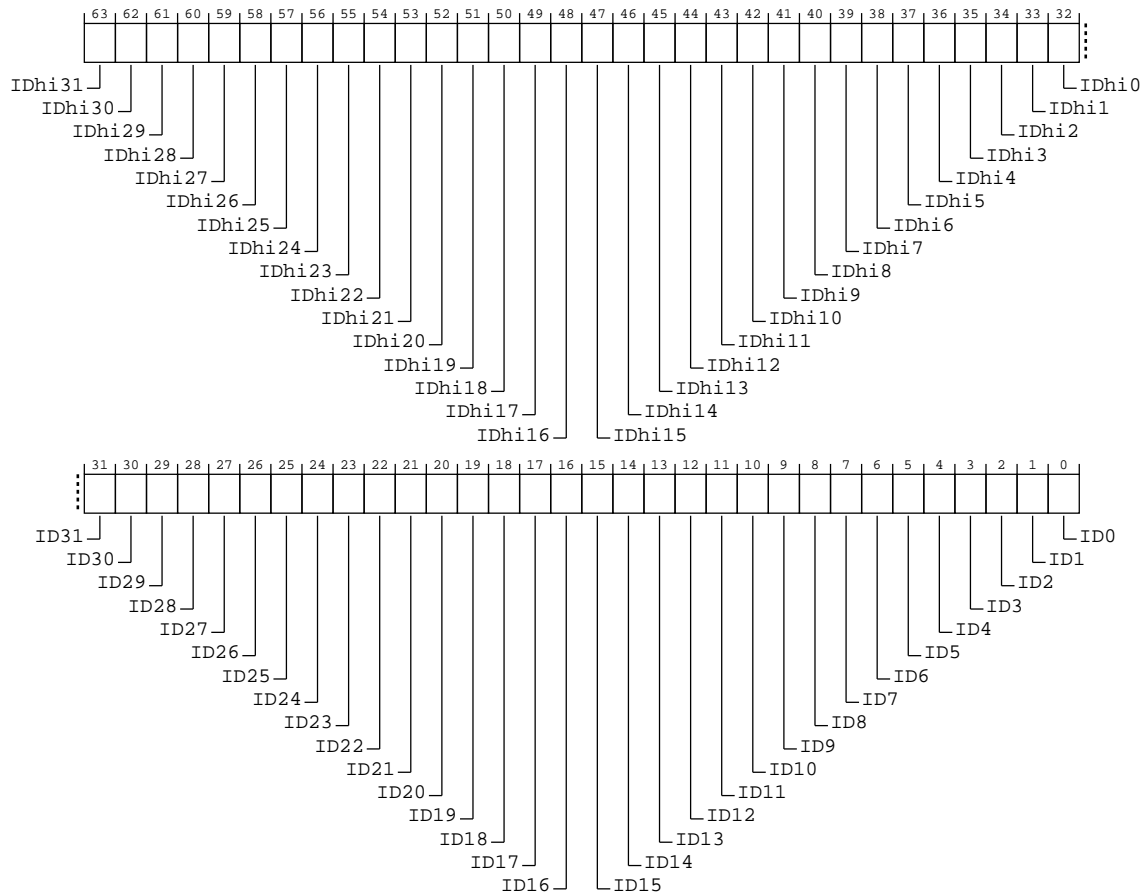
Performance Monitors

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-77: AArch64\_pmceid1\_el0 bit assignments**



**Table A-214: PMCEID1\_ELO bit descriptions**

Bits	Name	Description	Reset
[63]	IDHi31	IDHi31 corresponds to a Reserved Event event (0x403f) <b>0b0</b> The common event is not implemented, or not counted.	
[62]	IDHi30	IDHi30 corresponds to a Reserved Event event (0x403e) <b>0b0</b> The common event is not implemented, or not counted.	
[61]	IDHi29	IDHi29 corresponds to a Reserved Event event (0x403d) <b>0b0</b> The common event is not implemented, or not counted.	
[60]	IDHi28	IDHi28 corresponds to a Reserved Event event (0x403c) <b>0b0</b> The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[59]	IDHi27	IDHi27 corresponds to a Reserved Event event (0x403b) <b>0b0</b> The common event is not implemented, or not counted.	
[58]	IDHi26	IDHi26 corresponds to a Reserved Event event (0x403a) <b>0b0</b> The common event is not implemented, or not counted.	
[57]	IDHi25	IDHi25 corresponds to a Reserved Event event (0x4039) <b>0b0</b> The common event is not implemented, or not counted.	
[56]	IDHi24	IDHi24 corresponds to a Reserved Event event (0x4038) <b>0b0</b> The common event is not implemented, or not counted.	
[55]	IDHi23	IDHi23 corresponds to a Reserved Event event (0x4037) <b>0b0</b> The common event is not implemented, or not counted.	
[54]	IDHi22	IDHi22 corresponds to a Reserved Event event (0x4036) <b>0b0</b> The common event is not implemented, or not counted.	
[53]	IDHi21	IDHi21 corresponds to a Reserved Event event (0x4035) <b>0b0</b> The common event is not implemented, or not counted.	
[52]	IDHi20	IDHi20 corresponds to a Reserved Event event (0x4034) <b>0b0</b> The common event is not implemented, or not counted.	
[51]	IDHi19	IDHi19 corresponds to a Reserved Event event (0x4033) <b>0b0</b> The common event is not implemented, or not counted.	
[50]	IDHi18	IDHi18 corresponds to a Reserved Event event (0x4032) <b>0b0</b> The common event is not implemented, or not counted.	
[49]	IDHi17	IDHi17 corresponds to a Reserved Event event (0x4031) <b>0b0</b> The common event is not implemented, or not counted.	
[48]	IDHi16	IDHi16 corresponds to a Reserved Event event (0x4030) <b>0b0</b> The common event is not implemented, or not counted.	
[47]	IDHi15	IDHi15 corresponds to a Reserved Event event (0x402f) <b>0b0</b> The common event is not implemented, or not counted.	
[46]	IDHi14	IDHi14 corresponds to a Reserved Event event (0x402e) <b>0b0</b> The common event is not implemented, or not counted.	



Bits	Name	Description	Reset
[45]	IDHi13	IDHi13 corresponds to a Reserved Event event (0x402d) <b>0b0</b> The common event is not implemented, or not counted.	
[44]	IDHi12	IDHi12 corresponds to a Reserved Event event (0x402c) <b>0b0</b> The common event is not implemented, or not counted.	
[43]	IDHi11	IDHi11 corresponds to a Reserved Event event (0x402b) <b>0b0</b> The common event is not implemented, or not counted.	
[42]	IDHi10	IDHi10 corresponds to a Reserved Event event (0x402a) <b>0b0</b> The common event is not implemented, or not counted.	
[41]	IDHi9	IDHi9 corresponds to a Reserved Event event (0x4029) <b>0b0</b> The common event is not implemented, or not counted.	
[40]	IDHi8	IDHi8 corresponds to a Reserved Event event (0x4028) <b>0b0</b> The common event is not implemented, or not counted.	
[39]	IDHi7	IDHi7 corresponds to a Reserved Event event (0x4027) <b>0b0</b> The common event is not implemented, or not counted.	
[38]	IDHi6	IDHi6 corresponds to common event (0x4026) MEM_ACCESS_CHECKED_WR <b>0b1</b> The common event is implemented.	
[37]	IDHi5	IDHi5 corresponds to common event (0x4025) MEM_ACCESS_CHECKED_RD <b>0b1</b> The common event is implemented.	
[36]	IDHi4	IDHi4 corresponds to common event (0x4024) MEM_ACCESS_CHECKED <b>0b1</b> The common event is implemented.	
[35]	IDHi3	IDHi3 corresponds to common event (0x4023) Reserved <b>0b0</b> The common event is not implemented, or not counted.	
[34]	IDHi2	IDHi2 corresponds to common event (0x4022) ST_ALIGN_LAT <b>0b1</b> The common event is implemented.	
[33]	IDHi1	IDHi1 corresponds to common event (0x4021) LD_ALIGN_LAT <b>0b1</b> The common event is implemented.	
[32]	IDHi0	IDHi0 corresponds to common event (0x4020) LDST_ALIGN_LAT <b>0b1</b> The common event is implemented.	

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x3f) STALL_SLOT <b>0b1</b> The common event is implemented.	
[30]	ID30	ID30 corresponds to common event (0x3e) STALL_SLOT_FRONTEND <b>0b1</b> The common event is implemented.	
[29]	ID29	ID29 corresponds to common event (0x3d) STALL_SLOT_BACKEND <b>0b1</b> The common event is implemented.	
[28]	ID28	ID28 corresponds to common event (0x3c) STALL <b>0b1</b> The common event is implemented.	
[27]	ID27	ID27 corresponds to common event (0x3b) OP_SPEC <b>0b1</b> The common event is implemented.	
[26]	ID26	ID26 corresponds to common event (0x3a) OP_RETIRED <b>0b1</b> The common event is implemented.	
[25]	ID25	ID25 corresponds to common event (0x39) L1D_CACHE_LMISS_RD <b>0b1</b> The common event is implemented.	
[24]	ID24	ID24 corresponds to common event (0x38) REMOTE_ACCESS_RD <b>0b0</b> The common event is not implemented, or not counted.	
[23]	ID23	ID23 corresponds to common event (0x37) LL_CACHE_MISS_RD <b>0b1</b> The common event is implemented.	
[22]	ID22	ID22 corresponds to common event (0x36) LL_CACHE_RD <b>0b1</b> The common event is implemented.	
[21]	ID21	ID21 corresponds to common event (0x35) ITLB_WALK <b>0b1</b> The common event is implemented.	
[20]	ID20	ID20 corresponds to common event (0x34) DTLB_WALK <b>0b1</b> The common event is implemented.	
[19]	ID19	ID19 corresponds to a Reserved Event event (0x33) <b>0b0</b> The common event is not implemented, or not counted.	
[18]	ID18	ID18 corresponds to a Reserved Event event (0x32) <b>0b0</b> The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[17]	ID17	ID17 corresponds to common event (0x31) REMOTE_ACCESS <b>0b1</b> The common event is implemented.	
[16]	ID16	ID16 corresponds to common event (0x30) L2I_TLB <b>0b0</b> The common event is not implemented, or not counted.	
[15]	ID15	ID15 corresponds to common event (0x2f) L2D_TLB <b>0b1</b> The common event is implemented.	
[14]	ID14	ID14 corresponds to common event (0x2e) L2I_TLB_REFILL <b>0b0</b> The common event is not implemented, or not counted.	
[13]	ID13	ID13 corresponds to common event (0x2d) L2D_TLB_REFILL <b>0b1</b> The common event is implemented.	
[12]	ID12	ID12 corresponds to common event (0x2c) Reserved <b>0b0</b> The common event is not implemented, or not counted.	
[11]	ID11	ID11 corresponds to common event (0x2b) L3D_CACHE <b>0b1</b> The common event is implemented.	
[10]	ID10	ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL <b>0b1</b> The common event is implemented.	
[9]	ID9	ID9 corresponds to common event (0x29) L3D_CACHE_ALLOCATE <b>0b1</b> The common event is implemented.	
[8]	ID8	ID8 corresponds to common event (0x28) L2I_CACHE_REFILL <b>0b0</b> The common event is not implemented, or not counted.	
[7]	ID7	ID7 corresponds to common event (0x27) L2I_CACHE <b>0b0</b> The common event is not implemented, or not counted.	
[6]	ID6	ID6 corresponds to common event (0x26) L1I_TLB <b>0b1</b> The common event is implemented.	
[5]	ID5	ID5 corresponds to common event (0x25) L1D_TLB <b>0b1</b> The common event is implemented.	
[4]	ID4	ID4 corresponds to common event (0x24) STALL_BACKEND <b>0b1</b> The common event is implemented.	

Bits	Name	Description	Reset
[3]	ID3	ID3 corresponds to common event (0x23) STALL_FRONTEND  <b>0b1</b> The common event is implemented.	
[2]	ID2	ID2 corresponds to common event (0x22) BR_MIS_PRED_RETIRED  <b>0b1</b> The common event is implemented.	
[1]	ID1	ID1 corresponds to common event (0x21) BR_RETIRED  <b>0b1</b> The common event is implemented.	
[0]	ID0	ID0 corresponds to common event (0x20) L2D_CACHE_ALLOCATE  <b>0b1</b> The common event is implemented.	

## Access

MRS <Xt>, PMCEID1\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
PMCEID1_ELO	0b11	0b011	0b1001	0b1100	0b111

## Accessibility

MRS <Xt>, PMCEID1\_ELO

```

if PSTATE.EL == EL0 then
    if PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCEID1_ELO;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && MDCR_EL2.TPM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCEID1_ELO;
    elsif PSTATE.EL == EL2 then
        if MDCR_EL3.TPM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return PMCEID1_ELO;
    elsif PSTATE.EL == EL3 then
        return PMCEID1_ELO;

```

## A.6 AArch64 GIC register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** GIC registers in the core. Individual register descriptions provide detailed information.

**Table A-216: GIC register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
ICC_CTLR_EL1	3	C12	0	C12	4	See individual bit resets.	64-bit	Interrupt Controller Control Register (EL1)
ICV_CTLR_EL1	3	C12	0	C12	4	See individual bit resets.	64-bit	Interrupt Controller Virtual Control Register
ICC_AP0R0_EL1	3	C12	0	C8	4	See individual bit resets.	64-bit	Interrupt Controller Active Priorities Group 0 Registers
ICV_AP0R0_EL1	3	C12	0	C8	4	See individual bit resets.	64-bit	Interrupt Controller Virtual Active Priorities Group 0 Registers
ICC_AP1R0_EL1	3	C12	0	C9	0	See individual bit resets.	64-bit	Interrupt Controller Active Priorities Group 1 Registers
ICV_AP1R0_EL1	3	C12	0	C9	0	See individual bit resets.	64-bit	Interrupt Controller Virtual Active Priorities Group 1 Registers
ICH_VTR_EL2	3	C12	4	C11	1	See individual bit resets.	64-bit	Interrupt Controller VGIC Type Register
ICC_CTLR_EL3	3	C12	6	C12	4	See individual bit resets.	64-bit	Interrupt Controller Control Register (EL3)

### A.6.1 ICC\_CTLR\_EL1, Interrupt Controller Control Register (EL1)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

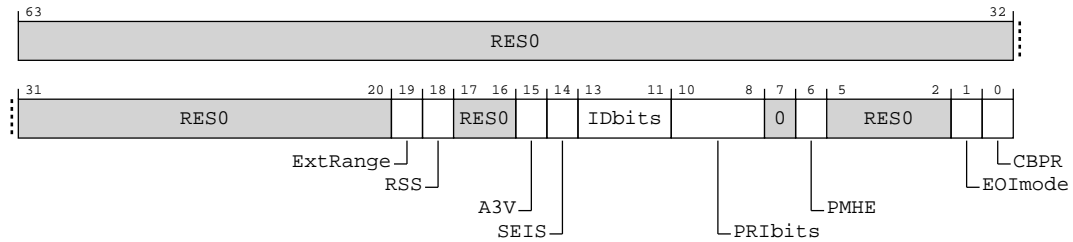
GIC

##### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-78: AArch64\_icc\_ctlr\_el1 bit assignments**



**Table A-217: ICC\_CTLR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	0x0
[19]	ExtRange	Extended INTID range (read-only). <b>0b1</b> CPU interface supports INTIDs in the range 1024..8191 <ul style="list-style-type: none"> <li>All INTIDs in the range 1024..8191 are treated as requiring deactivation.</li> </ul>	
[18]	RSS	Range Selector Support. Possible values are: <b>0b0</b> Targeted SGIs with affinity level 0 values of 0 - 15 are supported.	
[17:16]	RES0	Reserved	0b00
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. Possible values are: <b>0b1</b> The CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.	
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs: <b>0b0</b> The CPU interface logic does not support local generation of SEIs.	
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported: <b>0b000</b> 16 bits.	

Bits	Name	Description	Reset
[10:8]	PRIbits	<p>Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.</p> <p>An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).</p> <p>An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).</p> <p><b>Note:</b> This field always returns the number of priority bits implemented, regardless of the Security state of the access or the value of ext-GICD_CTLR.DS. For physical accesses, this field determines the minimum value of AArch64-ICC_BPR0_EL1.</p> <p>If EL3 is implemented, physical accesses return the value from AArch64-ICC_CTLR_EL3.PRIbits.</p> <p><b>0b100</b> 5 bits of priority are implemented</p>	
[7]	RES0	Reserved	0b0
[6]	PMHE	<p>Priority Mask Hint Enable. Controls whether the priority mask register is used as a hint for interrupt distribution:</p> <p><b>0b0</b> Disables use of AArch64-ICC_PMR_EL1 as a hint for interrupt distribution.</p> <p><b>0b1</b> Enables use of AArch64-ICC_PMR_EL1 as a hint for interrupt distribution.</p> <p>If EL3 is implemented, this bit is an alias of AArch64-ICC_CTLR_EL3.PMHE. Whether this bit can be written as part of an access to this register depends on the value of ext-GICD_CTLR.DS:</p> <ul style="list-style-type: none"> <li>If ext-GICD_CTLR.DS == 0, this bit is read-only.</li> <li>If ext-GICD_CTLR.DS == 1, this bit is read/write.</li> </ul>	
[5:2]	RES0	Reserved	0b0000
[1]	EOImode	<p>EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt:</p> <p><b>0b0</b> AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are UNPREDICTABLE.</p> <p><b>0b1</b> AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.</p> <p>The Secure AArch64-ICC_CTLR_EL1.EOImode is an alias of AArch64-ICC_CTLR_EL3.EOImode_EL1S.</p> <p>The Non-secure AArch64-ICC_CTLR_EL1.EOImode is an alias of AArch64-ICC_CTLR_EL3.EOImode_EL1NS.</p>	

Bits	Name	Description	Reset
[0]	CBPR	<p>Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupts:</p> <p><b>0b0</b></p> <p>AArch64-ICC_BPRO_EL1 determines the preemption group for Group 0 interrupts only.</p> <p>AArch64-ICC_BPR1_EL1 determines the preemption group for Group 1 interrupts.</p> <p><b>0b1</b></p> <p>AArch64-ICC_BPRO_EL1 determines the preemption group for both Group 0 and Group 1 interrupts.</p> <p>If EL3 is implemented:</p> <ul style="list-style-type: none"> <li>This bit is an alias of AArch64-ICC_CTLR_EL3.CBPR_EL1{S,NS} where S or NS corresponds to the current Security state.</li> <li>If ext-GICD_CTLR.DS == 0, this bit is read-only.</li> <li>If ext-GICD_CTLR.DS == 1, this bit is read/write.</li> </ul>	

## Access

MRS <Xt>, ICC\_CTLR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ICC_CTLR_EL1	0b11	0b000	0b1100	0b1100	0b100

MSR ICC\_CTLR\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_CTLR_EL1	0b11	0b000	0b1100	0b1100	0b100

## Accessibility

MRS <Xt>, ICC\_CTLR\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        return ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        return ICV_CTLR_EL1;
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.<IRQ,FIQ> == '11' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            if SCR_EL3.NS == '0' then
                return ICC_CTLR_EL1_S;
            else
                return ICC_CTLR_EL1_NS;
    elsif PSTATE.EL == EL3 then

```



```

if SCR_EL3.NS == '0' then
    return ICC_CTLR_EL1_S;
else
    return ICC_CTLR_EL1_NS;

```

MSR ICC\_CTLR\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elsif SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.<IRQ,FIQ> == '11' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t];
            else
                ICC_CTLR_EL1_NS = X[t];
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];

```

## A.6.2 ICV\_CTLR\_EL1, Interrupt Controller Virtual Control Register

Controls aspects of the behavior of the GIC virtual CPU interface and provides information about the features implemented.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

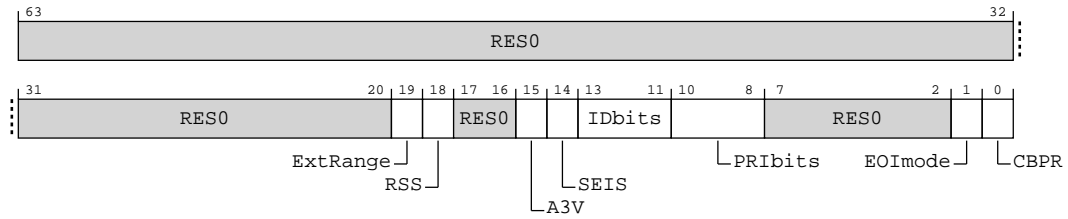
GIC

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-79: AArch64\_icv\_ctlr\_el1 bit assignments**



**Table A-220: ICV\_CTLR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	0x0
[19]	ExtRange	Extended INTID range (read-only). <b>0b1</b> CPU interface supports INTIDs in the range 1024..8191 <ul style="list-style-type: none"> <li>All INTIDs in the range 1024..8191 are treated as requiring deactivation.</li> </ul>	
[18]	RSS	Range Selector Support. Possible values are: <b>0b0</b> Targeted SGIs with affinity level 0 values of 0 - 15 are supported.	
[17:16]	RES0	Reserved	0b00
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. Possible values are: <b>0b1</b> The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.	
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the virtual CPU interface supports local generation of SEIs: <b>0b0</b> The virtual CPU interface logic does not support local generation of SEIs.	
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. The number of virtual interrupt identifier bits supported: <b>0b000</b> 16 bits.	
[10:8]	PRIbits	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).  <b>Note:</b> This field always returns the number of priority bits implemented. The division between group priority and subpriority is defined in the binary point registers AArch64-ICV_BPRO_EL1 and AArch64-ICV_BPR1_EL1. <b>0b100</b> 5 bits of priority are implemented	
[7:2]	RES0	Reserved	0b000000

Bits	Name	Description	Reset
[1]	EOImode	<p>Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt:</p> <p><b>0b0</b></p> <p>AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICV_DIR_EL1 are UNPREDICTABLE.</p> <p><b>0b1</b></p> <p>AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide priority drop functionality only. AArch64-ICV_DIR_EL1 provides interrupt deactivation functionality.</p>	
[0]	CBPR	<p>Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts:</p> <p><b>0b0</b></p> <p>AArch64-ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts.</p> <p><b>0b1</b></p> <p>Reads of AArch64-ICV_BPR1_EL1 return AArch64-ICV_BPR0_EL1 plus one, saturated to 0b111. Writes to AArch64-ICV_BPR1_EL1 are ignored.</p>	

## Access

MRS <Xt>, ICC\_CTLR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ICC_CTLR_EL1	0b11	0b000	0b1100	0b1100	0b100

MSR ICC\_CTLR\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_CTLR_EL1	0b11	0b000	0b1100	0b1100	0b100

## Accessibility

MRS <Xt>, ICC\_CTLR\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.FMO == '1' then
        return ICV_CTLR_EL1;
    elseif EL2Enabled() && HCR_EL2.IMO == '1' then
        return ICV_CTLR_EL1;
    elseif SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_CTLR_EL1_S;
        else
            return ICC_CTLR_EL1_NS;
    endif
elseif PSTATE.EL == EL2 then
    if SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            return ICC_CTLR_EL1_S;
        else

```

```

        return ICC_CTLR_EL1_NS;
elseif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        return ICC_CTLR_EL1_S;
    else
        return ICC_CTLR_EL1_NS;

```

MSR ICC\_CTLR\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elseif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t];
    elseif SCR_EL3.<IRQ,FIQ> == '11' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];
    elseif PSTATE.EL == EL2 then
        if SCR_EL3.<IRQ,FIQ> == '11' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t];
            else
                ICC_CTLR_EL1_NS = X[t];
    elseif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            ICC_CTLR_EL1_S = X[t];
        else
            ICC_CTLR_EL1_NS = X[t];

```

## A.6.3 ICC\_AP0R0\_EL1, Interrupt Controller Active Priorities Group 0 Registers

Provides information about Group 0 active priorities.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

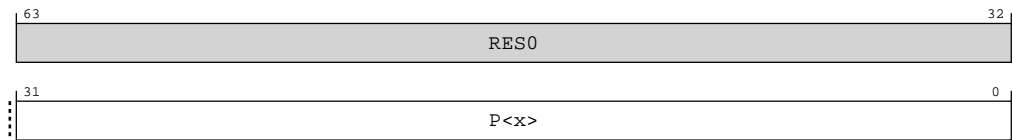
GIC

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-80: AArch64\_icc\_ap0r0\_el1 bit assignments**



**Table A-223: ICC\_AP0R0\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:0]	P<x>	Provides the access to the active priorities for Group 0 interrupts. Possible values of each bit are:  <b>0b0</b> 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.  <b>0b1</b> There is a Group 0 interrupt active with this priority level which has not undergone priority drop.  There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

### Access

MRS <Xt>, ICC\_AP0R0\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP0R0_EL1	0b11	0b000	0b1100	0b1000	0b100

MSR ICC\_AP0R0\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP0R0_EL1	0b11	0b000	0b1100	0b1000	0b100

## A.6.4 ICV\_AP0R0\_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers

Provides information about virtual Group 0 active priorities.

### Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC

Reset value

See individual bit resets.

Bit descriptions

Figure A-81: AArch64\_icv\_ap0r0\_el1 bit assignments

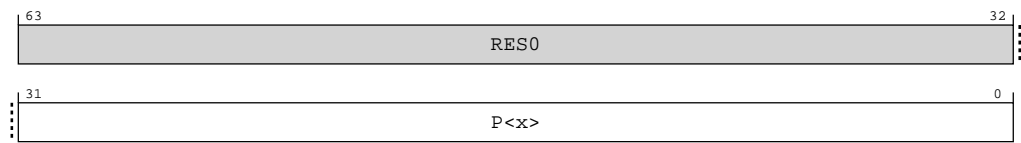


Table A-226: ICV\_AP0R0\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:0]	P<x>	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are:  <b>0b0</b> 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.  <b>0b1</b> There is a Group 0 interrupt active with this priority level which has not undergone priority drop.  There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Access

MRS <Xt>, ICC\_AP0R0\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP0R0_EL1	0b11	0b000	0b1100	0b1000	0b100

MSR ICC\_AP0R0\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP0R0_EL1	0b11	0b000	0b1100	0b1000	0b100

### A.6.5 ICC\_AP1R0\_EL1, Interrupt Controller Active Priorities Group 1 Registers

Provides information about Group 1 active priorities.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

GIC

##### Reset value

See individual bit resets.

#### Bit descriptions

Figure A-82: AArch64\_icc\_ap1r0\_el1 bit assignments

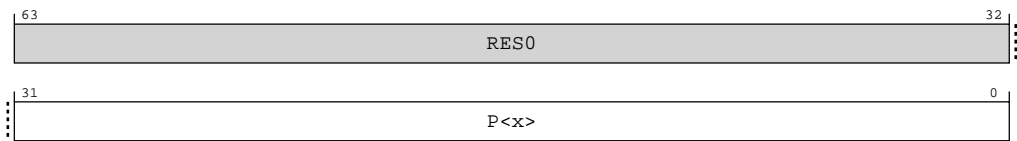


Table A-229: ICC\_AP1R0\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:0]	P<x>	<p>Group 1 interrupt active priorities. When AArch64-SCR_EL3.NS == '1', accesses the priorities for Non-secure Group 1 interrupts, and when AArch64-SCR_EL3.NS == '0' accesses the priorities for Secure Group 1 interrupts. Possible values of each bit are:</p> <p><b>0b0</b></p> <p>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.</p> <p><b>0b1</b></p> <p>There is a Group 1 interrupt active with this priority level which has not undergone priority drop.</p> <p>There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].</p> <p>When accessed from non-secure EL2 or EL1, only the 16 lowest-priority interrupts are visible in bits [15:0] of this register.</p>	

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

**Access**

MRS &lt;Xt&gt;, ICC\_AP1R0\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP1R0_EL1	0b11	0b000	0b1100	0b1001	0b000

MSR ICC\_AP1R0\_EL1, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP1R0_EL1	0b11	0b000	0b1100	0b1001	0b000

## A.6.6 ICV\_AP1R0\_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers

Provides information about virtual Group 1 active priorities.

**Configurations**

This register is available in all configurations.

**Attributes****Width**

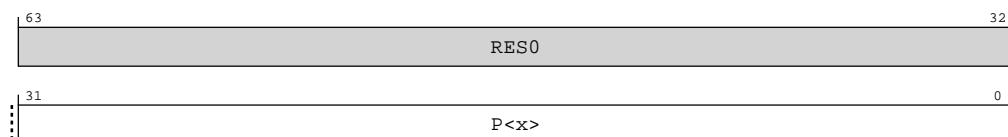
64

**Functional group**

GIC

**Reset value**

See individual bit resets.

**Bit descriptions****Figure A-83: AArch64\_icv\_ap1r0\_el1 bit assignments****Table A-232: ICV\_AP1R0\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0



Bits	Name	Description	Reset
[31:0]	P<x>	<p>Group 1 interrupt active priorities. Possible values of each bit are:</p> <p><b>0b0</b></p> <p>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.</p> <p><b>0b1</b></p> <p>There is a Group 1 interrupt active with this priority level which has not undergone priority drop.</p> <p>There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].</p>	

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

### Access

MRS <Xt>, ICC\_AP1R0\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP1R0_EL1	0b11	0b000	0b1100	0b1001	0b000

MSR ICC\_AP1R0\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_AP1R0_EL1	0b11	0b000	0b1100	0b1001	0b000

## A.6.7 ICH\_VTR\_EL2, Interrupt Controller VGIC Type Register

Reports supported GIC virtualization features.

### Configurations

If EL2 is not implemented, all bits in this register are RES0 from EL3, except for nV4, which is RES1 from EL3.

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

### Attributes

#### Width

64

#### Functional group

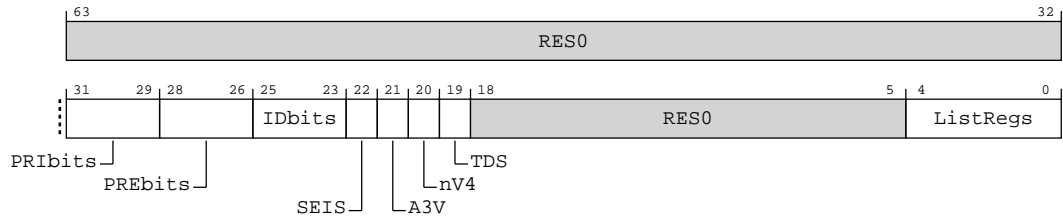
GIC

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-84: AArch64\_ich\_vtr\_el2 bit assignments**



**Table A-235: ICH\_VTR\_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:29]	PRIbits	<p>Priority bits. The number of virtual priority bits implemented, minus one.</p> <p>An implementation must implement at least 32 levels of virtual priority (5 priority bits).</p> <p>This field is an alias of AArch64-ICV_CTLR_EL1.PRIbits.</p> <p><b>0b100</b></p> <p>5 virtual priority bits are implemented</p>	
[28:26]	PREbits	<p>The number of virtual preemption bits implemented, minus one.</p> <p>An implementation must implement at least 32 levels of virtual preemption priority (5 preemption bits).</p> <p>The value of this field must be less than or equal to the value of ICH_VTR_EL2.PRIbits.</p> <p>The maximum value of this field is 6, indicating 7 bits of preemption.</p> <p>This field determines the minimum value of AArch64-ICH_VMCR_EL2.VBPR0.</p> <p><b>0b100</b></p> <p>5 virtual pre-emption bits are implemented</p>	
[25:23]	IDbits	<p>The number of virtual interrupt identifier bits supported:</p> <p><b>0b000</b></p> <p>16 bits.</p>	
[22]	SEIS	<p>SEI Support. Indicates whether the virtual CPU interface supports generation of SEIs:</p> <p><b>0b0</b></p> <p>The virtual CPU interface logic does not support generation of SEIs.</p>	
[21]	A3V	<p>Affinity 3 Valid. Possible values are:</p> <p><b>0b1</b></p> <p>The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.</p>	
[20]	nV4	<p>Direct injection of virtual interrupts not supported. Possible values are:</p> <p><b>0b0</b></p> <p>The CPU interface logic supports direct injection of virtual interrupts.</p>	

Bits	Name	Description	Reset
[19]	TDS	Separate trapping of EL1 writes to AArch64-ICV_DIR_EL1 supported.  <b>0b1</b> Implementation supports AArch64-ICH_HCR_EL2.TDIR.	
[18:5]	RES0	Reserved	0x0
[4:0]	ListRegs	The number of implemented List registers, minus one. For example, a value of 0b01111 indicates that the maximum of 16 List registers are implemented.  <b>0b00011</b> 4 List registers	

### Access

MRS <Xt>, ICH\_VTR\_EL2

<systemreg>	op0	op1	CRn	CRm	op2
ICH_VTR_EL2	0b11	0b100	0b1100	0b1011	0b001

### Accessibility

MRS <Xt>, ICH\_VTR\_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    return ICH_VTR_EL2;
elseif PSTATE.EL == EL3 then
    return ICH_VTR_EL2;

```

## A.6.8 ICC\_CTLR\_EL3, Interrupt Controller Control Register (EL3)

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

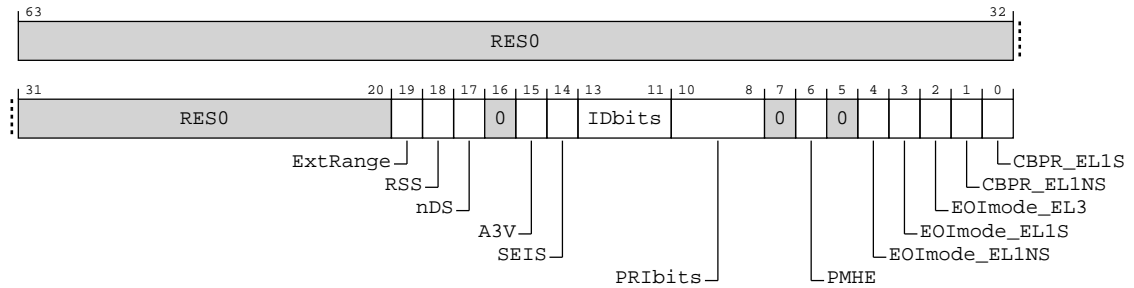
GIC

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-85: AArch64\_icc\_ctlr\_el3 bit assignments**



**Table A-237: ICC\_CTLR\_EL3 bit descriptions**

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	0x0
[19]	ExtRange	Extended INTID range (read-only). <b>0b1</b> CPU interface supports INTIDs in the range 1024..8191 <ul style="list-style-type: none"> <li>All INTIDs in the range 1024..8191 are treated as requiring deactivation.</li> </ul>	
[18]	RSS	Range Selector Support. <b>0b0</b> Targeted SGIs with affinity level 0 values of 0-15 are supported.	
[17]	nDS	Disable Security not supported. Read-only and writes are ignored. <b>0b1</b> The CPU interface logic does not support disabling of security, and requires that security is not disabled.	
[16]	RES0	Reserved	0x0
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. <b>0b1</b> The CPU interface logic supports non-zero values of the Aff3 field in SGI generation System registers.	
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports generation of SEIs: <b>0b0</b> The CPU interface logic does not support generation of SEIs.	
[13:11]	IDbits	Identifier bits. Read-only and writes are ignored. Indicates the number of physical interrupt identifier bits supported. <b>0b000</b> 16 bits.	

Bits	Name	Description	Reset
[10:8]	PRIbits	<p>Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.</p> <p>An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).</p> <p>An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).</p> <p><b>Note:</b> This field always returns the number of priority bits implemented, regardless of the value of SCR_EL3.NS or the value of ext-GICD_CTLR.DS. The division between group priority and subpriority is defined in the binary point registers AArch64-ICC_BPR0_EL1 and AArch64-ICC_BPR1_EL1.</p> <p>This field determines the minimum value of ICC_BPR0_EL1.</p> <p><b>0b100</b> 5 bits of priority are implemented</p>	
[7]	RES0	Reserved	0b0
[6]	PMHE	<p>Priority Mask Hint Enable.</p> <p><b>0b0</b> Disables use of the priority mask register as a hint for interrupt distribution.</p> <p><b>0b1</b> Enables use of the priority mask register as a hint for interrupt distribution.</p> <p>Software must write AArch64-ICC_PMR_EL1 to 0xFF before clearing this field to 0.</p> <ul style="list-style-type: none"> <li>An implementation might choose to make this field RAO/WI if priority-based routing is always used</li> <li>An implementation might choose to make this field RAZ/WI if priority-based routing is never used</li> </ul> <p>If EL3 is present, AArch64-ICC_CTLR_EL1.PMHE is an alias of ICC_CTLR_EL3.PMHE.</p>	0b0
[5]	RES0	Reserved	0b0
[4]	EOImode_EL1NS	<p>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.</p> <p><b>0b0</b> AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are UNPREDICTABLE.</p> <p><b>0b1</b> AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.</p> <p>If EL3 is present, AArch64-ICC_CTLR_EL1(NS).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1NS.</p>	

Bits	Name	Description	Reset
[3]	EOImode_EL1S	<p>EOI mode for interrupts handled at Secure EL1. Controls whether a write to an End of Interrupt register also deactivates the interrupt.</p> <p><b>0b0</b></p> <p>AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are UNPREDICTABLE.</p> <p><b>0b1</b></p> <p>AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.</p> <p>If EL3 is present, AArch64-ICC_CTLR_EL1(S).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1S.</p>	
[2]	EOImode_EL3	<p>EOI mode for interrupts handled at EL3. Controls whether a write to an End of Interrupt register also deactivates the interrupt.</p> <p><b>0b0</b></p> <p>AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to AArch64-ICC_DIR_EL1 are UNPREDICTABLE.</p> <p><b>0b1</b></p> <p>AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.</p>	
[1]	CBPR_EL1NS	<p>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.</p> <p><b>0b0</b></p> <p>AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only.</p> <p>AArch64-ICC_BPR1_EL1 determines the preemption group for Non-secure Group 1 interrupts.</p> <p><b>0b1</b></p> <p>AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts and Non-secure Group 1 interrupts. Non-secure accesses to ext-GICC_BPR and AArch64-ICC_BPR1_EL1 access the state of AArch64-ICC_BPR0_EL1.</p> <p>If EL3 is present, AArch64-ICC_CTLR_EL1(NS).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1NS.</p>	
[0]	CBPR_EL1S	<p>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.</p> <p><b>0b0</b></p> <p>AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only.</p> <p>AArch64-ICC_BPR1_EL1 determines the preemption group for Secure Group 1 interrupts.</p> <p><b>0b1</b></p> <p>AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts and Secure Group 1 interrupts. Secure EL1 accesses to AArch64-ICC_BPR1_EL1 access the state of AArch64-ICC_BPR0_EL1.</p> <p>If EL3 is present, AArch64-ICC_CTLR_EL1(S).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1S.</p>	

## Access

MRS <Xt>, ICC\_CTLR\_EL3

<systemreg>	op0	op1	CRn	CRm	op2
ICC_CTLR_EL3	0b11	0b110	0b1100	0b1100	0b100

MSR ICC\_CTLR\_EL3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ICC_CTLR_EL3	0b11	0b110	0b1100	0b1100	0b100

## Accessibility

MRS <Xt>, ICC\_CTLR\_EL3

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    return ICC_CTLR_EL3;
```

MSR ICC\_CTLR\_EL3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    UNDEFINED;
elseif PSTATE.EL == EL2 then
    UNDEFINED;
elseif PSTATE.EL == EL3 then
    ICC_CTLR_EL3 = X[t];
```

## A.7 AArch64 Activity Monitors register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Activity Monitors registers in the core. Individual register descriptions provide detailed information.

**Table A-240: Activity Monitors register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
AMEVTYPER10_ELO	3	C13	3	C14	0	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_ELO	3	C13	3	C14	1	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_ELO	3	C13	3	C14	2	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 1
AMCFGR_ELO	3	C13	3	C2	1	See individual bit resets.	64-bit	Activity Monitors Configuration Register
AMCGCR_ELO	3	C13	3	C2	2	See individual bit resets.	64-bit	Activity Monitors Counter Group Configuration Register

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
AMEVTYPER00_ELO	3	C13	3	C6	0	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_ELO	3	C13	3	C6	1	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_ELO	3	C13	3	C6	2	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_ELO	3	C13	3	C6	3	See individual bit resets.	64-bit	Activity Monitors Event Type Registers 0

### A.7.1 AMEVTYPER10\_ELO, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR10\_ELO counts.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

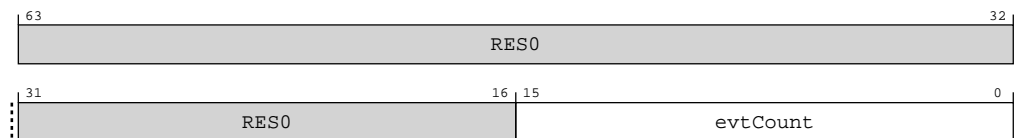
Activity Monitors

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure A-86: AArch64\_amevtyper10\_el0 bit assignments**



**Table A-241: AMEVTYPER10\_ELO bit descriptions**

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0



Bits	Name	Description	Reset
[15:0]	evtCount	<p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AArch64-AMEVCNTR1&lt;n&gt;_ELO.</p> <p>It is <b>IMPLEMENTATION DEFINED</b> what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter AArch64-AMEVCNTR1&lt;n&gt;_ELO, then:</p> <ul style="list-style-type: none"> <li>It is UNPREDICTABLE which event will be counted.</li> <li>The value read back is UNKNOWN.</li> </ul> <p>The event counted by AArch64-AMEVCNTR1&lt;n&gt;_ELO might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED.</p> <p>If the corresponding counter AArch64-AMEVCNTR1&lt;n&gt;_ELO is enabled, writes to this register have UNPREDICTABLE results.</p>	

### Access

MRS &lt;Xt&gt;, AMEVTYPEPER10\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPEPER10_ELO	0b11	0b011	0b1101	0b1110	0b000

MSR AMEVTYPEPER10\_ELO, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPEPER10_ELO	0b11	0b011	0b1101	0b1110	0b000

## A.7.2 AMEVTYPEPER11\_ELO, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR11\_ELO counts.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

Activity Monitors

#### Reset value

See individual bit resets.

Bit descriptions

Figure A-87: AArch64\_amevtyper11\_el0 bit assignments

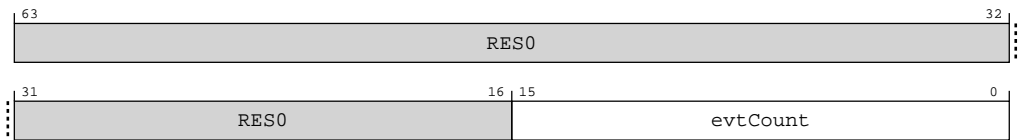


Table A-244: AMEVTYPER11\_ELO bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15:0]	evtCount	<p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AArch64-AMEVCNTR1&lt;n&gt;_ELO.</p> <p>It is <b>IMPLEMENTATION DEFINED</b> what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter AArch64-AMEVCNTR1&lt;n&gt;_ELO, then:</p> <ul style="list-style-type: none"><li>It is <b>UNPREDICTABLE</b> which event will be counted.</li><li>The value read back is <b>UNKNOWN</b>.</li></ul> <p>The event counted by AArch64-AMEVCNTR1&lt;n&gt;_ELO might be fixed at implementation. In this case, the field is read-only and writes are <b>UNDEFINED</b>.</p> <p>If the corresponding counter AArch64-AMEVCNTR1&lt;n&gt;_ELO is enabled, writes to this register have <b>UNPREDICTABLE</b> results.</p>	

Access

MRS <Xt>, AMEVTYPER11\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER11_ELO	0b11	0b011	0b1101	0b1110	0b001

MSR AMEVTYPER11\_ELO, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER11_ELO	0b11	0b011	0b1101	0b1110	0b001

A.7.3 AMEVTYPER12\_ELO, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR12\_ELO counts.

Configurations

This register is available in all configurations.

**Attributes****Width**

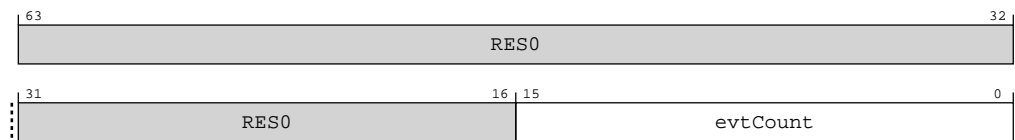
64

**Functional group**

Activity Monitors

**Reset value**

See individual bit resets.

**Bit descriptions****Figure A-88: AArch64\_amevtyper12\_el0 bit assignments****Table A-247: AMEVTYPER12\_ELO bit descriptions**

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15:0]	evtCount	<p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AArch64-AMEVCNTR1&lt;n&gt;_ELO.</p> <p>It is <b>IMPLEMENTATION DEFINED</b> what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter AArch64-AMEVCNTR1&lt;n&gt;_ELO, then:</p> <ul style="list-style-type: none"> <li>It is UNPREDICTABLE which event will be counted.</li> <li>The value read back is UNKNOWN.</li> </ul> <p>The event counted by AArch64-AMEVCNTR1&lt;n&gt;_ELO might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED.</p> <p>If the corresponding counter AArch64-AMEVCNTR1&lt;n&gt;_ELO is enabled, writes to this register have UNPREDICTABLE results.</p>	

**Access**

MRS &lt;Xt&gt;, AMEVTYPER12\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER12_ELO	0b11	0b011	0b1101	0b1110	0b010

MSR AMEVTYPER12\_ELO, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER12_ELO	0b11	0b011	0b1101	0b1110	0b010

## A.7.4 AMCFGR\_ELO, Activity Monitors Configuration Register

Global configuration register for the activity monitors.

Provides information on supported features, the number of counter groups implemented, the total number of activity monitor event counters implemented, and the size of the counters. AMCFGR\_ELO is applicable to both the architected and the auxiliary counter groups.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

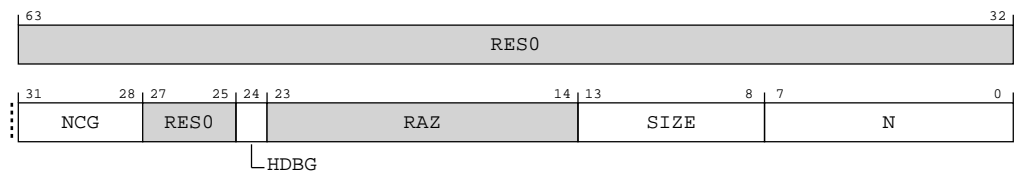
Activity Monitors

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-89: AArch64\_amcfgr\_el0 bit assignments**



**Table A-250: AMCFGR\_ELO bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:28]	NCG	Defines the number of counter groups. The following value is specified for this product. <b>0b0001</b> Two counter groups are implemented	
[27:25]	RES0	Reserved	0b000
[24]	HDBG	Halt-on-debug supported.  From Armv8, this feature must be supported, and so this bit is 0b1. <b>0b1</b> AArch64-AMCR_ELO.HDBG is read/write.	
[23:14]	RAZ	Reserved	

Bits	Name	Description	Reset
[13:8]	SIZE	<p>Defines the size of activity monitor event counters.</p> <p>The size of the activity monitor event counters implemented by the activity monitors Extension is defined as [AMCFGR_ELO.SIZE + 1].</p> <p>From Armv8, the counters are 64-bit, and so this field is 0b111111.</p> <p><b>Note:</b> Software also uses this field to determine the spacing of counters in the memory-map. From Armv8, the counters are at doubleword-aligned addresses.</p> <p><b>0b111111</b> 64 bits.</p>	
[7:0]	N	<p>Defines the number of activity monitor event counters.</p> <p>The total number of counters implemented in all groups by the Activity Monitors Extension is defined as [AMCFGR_ELO.N + 1].</p> <p><b>0b00000110</b> Seven activity monitor event counters</p>	

## Access

MRS <Xt>, AMCFGR\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
AMCFGR_ELO	0b11	0b011	0b1101	0b0010	0b001

## Accessibility

MRS <Xt>, AMCFGR\_ELO

```

if PSTATE.EL == EL0 then
    if AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elsif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif CPTR_EL3.TAM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return AMCFGR_EL0;
    elsif PSTATE.EL == EL1 then
        if EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif CPTR_EL3.TAM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return AMCFGR_EL0;
    elsif PSTATE.EL == EL2 then
        if CPTR_EL3.TAM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return AMCFGR_EL0;
    elsif PSTATE.EL == EL3 then
        return AMCFGR_EL0;

```

## A.7.5 AMCGCR\_ELO, Activity Monitors Counter Group Configuration Register

Provides information on the number of activity monitor event counters implemented within each counter group.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

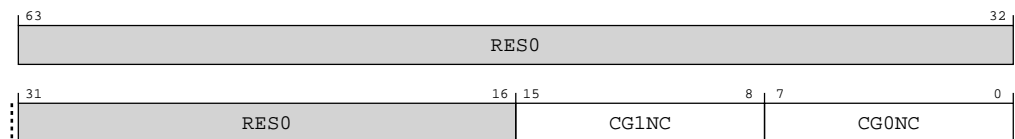
Activity Monitors

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-90: AArch64\_amcgr\_el0 bit assignments**



**Table A-252: AMCGCR\_ELO bit descriptions**

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15:8]	CG1NC	Counter Group 1 Number of Counters. The number of counters in the auxiliary counter group.  In AMUv1, the permitted range of values is 0x0 to 0x10.  <b>0b00000011</b> Three counters in the auxiliary counter group	
[7:0]	CG0NC	Counter Group 0 Number of Counters. The number of counters in the architected counter group.  In AMUv1, the value of this field is 0x4.  <b>0b00000100</b> Four Counters in the architected counter group	

### Access

MRS <Xt>, AMCGCR\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
AMCGCR_ELO	0b11	0b011	0b1101	0b0010	0b010

## Accessibility

MRS <Xt>, AMCGCR\_ELO

```

if PSTATE.EL == EL0 then
    if AMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
        elseif EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif CPTR_EL3.TAM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return AMCGCR_ELO;
    elseif PSTATE.EL == EL1 then
        if EL2Enabled() && CPTR_EL2.TAM == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elseif CPTR_EL3.TAM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return AMCGCR_ELO;
    elseif PSTATE.EL == EL2 then
        if CPTR_EL3.TAM == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return AMCGCR_ELO;
    elseif PSTATE.EL == EL3 then
        return AMCGCR_ELO;

```

## A.7.6 AMEVTYPER00\_ELO, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR00\_ELO counts.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

Activity Monitors

#### Reset value

See individual bit resets.

Bit descriptions

Figure A-91: AArch64\_amevtyper00\_el0 bit assignments

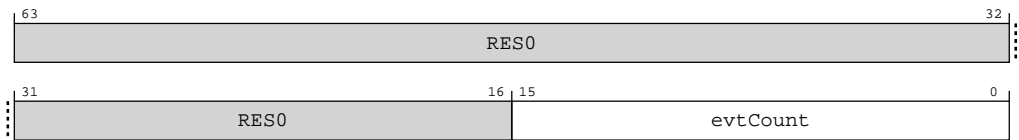


Table A-254: AMEVTYPER00\_ELO bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR00_ELO. The value of this field is architecturally required for each architected counter.	

Access

MRS <Xt>, AMEVTYPER00\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER00_ELO	0b11	0b011	0b1101	0b0110	0b000

A.7.7 AMEVTYPER01\_ELO, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR01\_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity Monitors

Reset value

See individual bit resets.



Bit descriptions

Figure A-92: AArch64\_amevtyper01\_el0 bit assignments

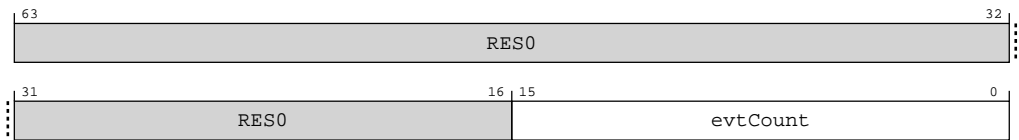


Table A-256: AMEVTYPER01\_ELO bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR01_ELO. The value of this field is architecturally required for each architected counter.	

Access

MRS <Xt>, AMEVTYPER01\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER01_ELO	0b11	0b011	0b1101	0b0110	0b001

A.7.8 AMEVTYPER02\_ELO, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR02\_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Activity Monitors

Reset value

See individual bit resets.

Bit descriptions

Figure A-93: AArch64\_amevtyper02\_el0 bit assignments

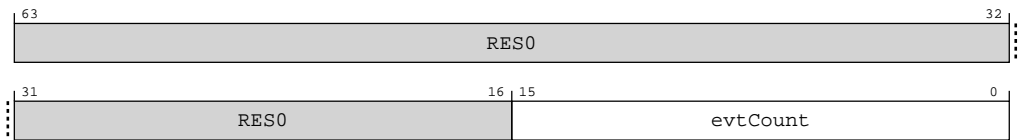


Table A-258: AMEVTYPER02\_ELO bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR02_ELO. The value of this field is architecturally required for each architected counter.	

**Access**  
MRS <Xt>, AMEVTYPER02\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER02_ELO	0b11	0b011	0b1101	0b0110	0b010

A.7.9 AMEVTYPER03\_ELO, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR03\_ELO counts.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

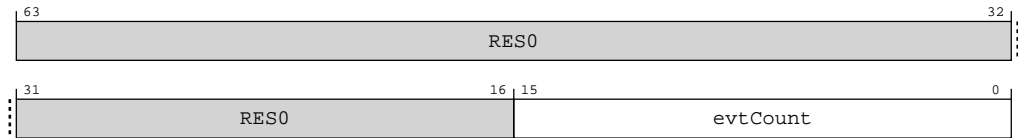
Activity Monitors

Reset value

See individual bit resets.

## Bit descriptions

**Figure A-94: AArch64\_amevtyper03\_el0 bit assignments**



**Table A-260: AMEVTPER03\_ELO bit descriptions**

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter AArch64-AMEVCNTR03_ELO. The value of this field is architecturally required for each architected counter.	

## Access

MRS <Xt>, AMEVTPER03\_ELO

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTPER03_ELO	0b11	0b011	0b1101	0b0110	0b011

## A.8 AArch64 Trace register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** Trace registers in the core. Individual register descriptions provide detailed information.

**Table A-262: Trace register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
TRCIDR8	2	C0	1	C0	6	See individual bit resets.	64-bit	ID Register 8
TRCIMSPECO	2	C0	1	C0	7	See individual bit resets.	64-bit	IMP DEF Register 0
TRCIDR2	2	C0	1	C10	7	See individual bit resets.	64-bit	ID Register 2
TRCIDR3	2	C0	1	C11	7	See individual bit resets.	64-bit	ID Register 3
TRCIDR4	2	C0	1	C12	7	See individual bit resets.	64-bit	ID Register 4
TRCIDR5	2	C0	1	C13	7	See individual bit resets.	64-bit	ID Register 5
TRCIDR10	2	C0	1	C2	6	0x0	64-bit	ID Register 10
TRCIDR11	2	C0	1	C3	6	0x0	64-bit	ID Register 11
TRCIDR12	2	C0	1	C4	6	0x0	64-bit	ID Register 12
TRCIDR13	2	C0	1	C5	6	0x0	64-bit	ID Register 13
TRCIDR0	2	C0	1	C8	7	See individual bit resets.	64-bit	ID Register 0
TRCIDR1	2	C0	1	C9	7	See individual bit resets.	64-bit	ID Register 1
TRCCIDCVRO	2	C3	1	C0	0	See individual bit resets.	64-bit	Context Identifier Comparator Value Registers <n>

## A.8.1 TRCIDR8, ID Register 8

Returns the maximum speculation depth of the instruction Trace element stream.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

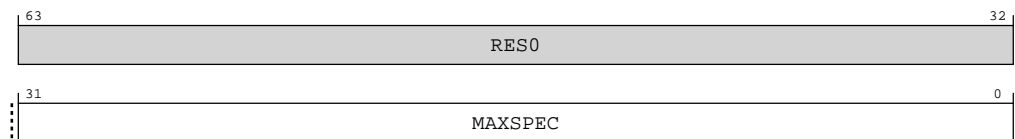
Trace

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-95: AArch64\_trcidr8 bit assignments**



**Table A-263: TRCIDR8 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:0]	MAXSPEC	Indicates the maximum speculation depth of the instruction Trace element stream. This is the maximum number of PO elements in the Trace element stream that can be speculative at any time.  0b00000000000000000000000000000000 No speculation in the Trace element stream	

### Access

MRS <Xt>, TRCIDR8

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR8	0b10	0b001	0b0000	0b0000	0b110

### Accessibility

MRS <Xt>, TRCIDR8

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then

```

```

if CPACR_EL1.TTA == '1' then
    AArch64.SystemAccessTrap(EL1, 0x18);
elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elsif CPTR_EL3.TTA == '1' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    return TRCIDR8;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR8;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR8;

```

## A.8.2 TRCIMSPECO, IMP DEF Register 0

TRCIMSPECO shows the presence of any **IMPLEMENTATION DEFINED** features, and provides an interface to enable the features that are provided.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

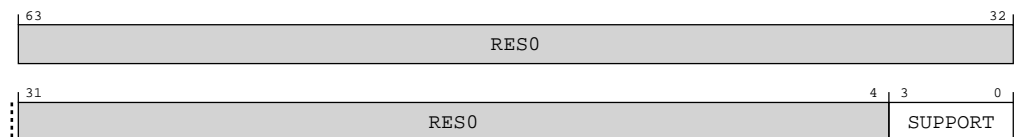
Trace

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-96: AArch64\_trcimspec0 bit assignments**



**Table A-265: TRCIMSPECO bit descriptions**

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	0x0
[3:0]	SUPPORT	Indicates whether the implementation supports <b>IMPLEMENTATION DEFINED</b> features.  <b>0b0000</b> No IMPLEMENTATION DEFINED features are supported.	

### Access

MRS &lt;Xt&gt;, TRCIMSPECO

<systemreg>	op0	op1	CRn	CRm	op2
TRCIMSPECO	0b10	0b001	0b0000	0b0000	0b111

MSR TRCIMSPECO, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
TRCIMSPECO	0b10	0b001	0b0000	0b0000	0b111

### Accessibility

MRS &lt;Xt&gt;, TRCIMSPECO

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCIMSPEcN == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPECO;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPECO;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPECO;

```

MSR TRCIMSPECO, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then

```

```

    AArch64.SystemAccessTrap(EL2, 0x18);
  elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDBGWTR_EL2.TRCIMSPECn == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
  elseif CPTR_EL3.TTA == '1' then
    AArch64.SystemAccessTrap(EL3, 0x18);
  else
    TRCIMSPECO = X[t];
  elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
      AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
      AArch64.SystemAccessTrap(EL3, 0x18);
    else
      TRCIMSPECO = X[t];
  elseif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
      AArch64.SystemAccessTrap(EL3, 0x18);
    else
      TRCIMSPECO = X[t];

```

### A.8.3 TRCIDR2, ID Register 2

Returns the tracing capabilities of the Trace unit.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

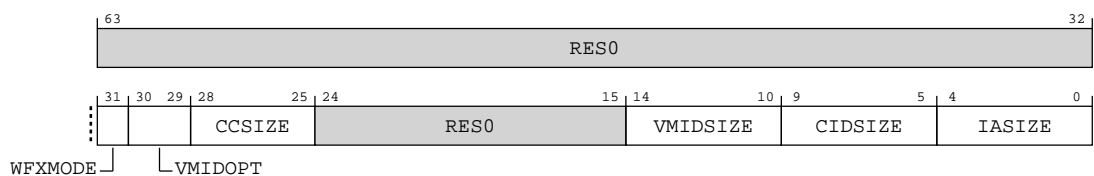
Trace

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure A-97: AArch64\_trcidr2 bit assignments**



**Table A-268: TRCIDR2 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[31]	WFXMODE	Indicates whether WFI and WFE instructions are classified as P0 instructions:  <b>0b1</b> WFI and WFE instructions are classified as P0 instructions.	
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection.  <b>0b10</b> Virtual context identifier selection not supported. AArch64-TRCCONFIGR.VMIDOPT is RES1.	
[28:25]	CCSIZE	Indicates the size of the cycle counter.  <b>0b0000</b> The cycle counter is 12 bits in length.	
[24:15]	RES0	Reserved	0x0
[14:10]	VMIDSIZE	Indicates the Trace unit Virtual context identifier size.  <b>0b00100</b> 32-bit Virtual context identifier size.	
[9:5]	CIDSIZE	Indicates the Context identifier size.  <b>0b00100</b> 32-bit Context identifier size.	
[4:0]	IASIZE	Virtual instruction address size.  <b>0b01000</b> Maximum of 64-bit instruction address size.	

## Access

MRS <Xt>, TRCIDR2

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR2	0b10	0b001	0b0000	0b1010	0b111

## Accessibility

MRS <Xt>, TRCIDR2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR2;
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return TRCIDR2;
    elsif PSTATE.EL == EL3 then

```



```
if CPTR_EL3.TTA == '1' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    return TRCIDR2;
```

A.8.4 TRCIDR3, ID Register 3

Returns the base architecture of the Trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

See individual bit resets.

Bit descriptions

Figure A-98: AArch64\_trcidr3 bit assignments

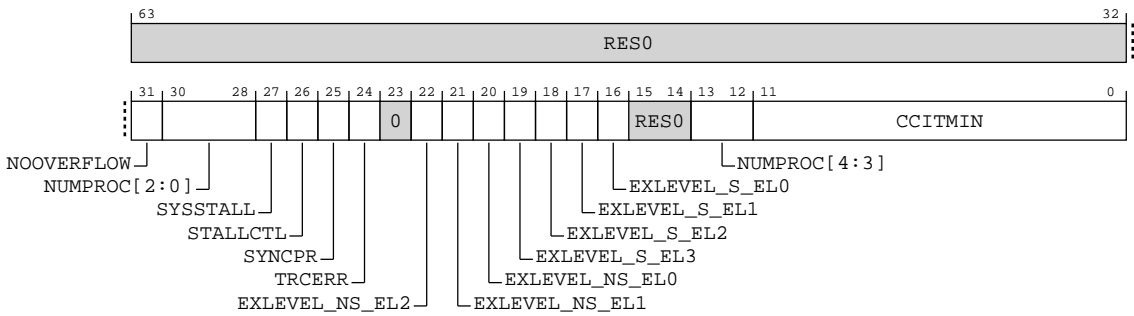


Table A-270: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented.  0b0 Overflow prevention is not implemented.	
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing.  0b00000 The Trace unit can Trace one PE.	

Bits	Name	Description	Reset
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. <b>0b0</b> Stalling of the PE is not permitted.	
[26]	STALLCTL	Indicates if Trace unit implements stalling of the PE. <b>0b0</b> Stalling of the PE is not implemented.	
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. <b>0b0</b> AArch64-TRCSYNCPR is read-write so software can change the synchronization period.	
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. <b>0b1</b> Forced tracing of System Error exceptions is implemented.	
[23]	RES0	Reserved	0x0
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 implemented. <b>0b1</b> Non-secure EL2 is implemented.	
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 implemented. <b>0b1</b> Non-secure EL1 is implemented.	
[20]	EXLEVEL_NS_ELO	Indicates if Non-secure ELO implemented. <b>0b1</b> Non-secure ELO is implemented.	
[19]	EXLEVEL_S_EL3	Indicates if Secure EL3 implemented. <b>0b1</b> Secure EL3 is implemented.	
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 implemented. <b>0b1</b> Secure EL2 is implemented.	
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 implemented. <b>0b1</b> Secure EL1 is implemented.	
[16]	EXLEVEL_S_ELO	Indicates if Secure ELO implemented. <b>0b1</b> Secure ELO is implemented.	
[15:14]	RES0	Reserved	0b00
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in AArch64-TRCCCCTLR.THRESHOLD.  If AArch64-TRCIDR0.TRCCCI == 0b1 then the minimum value of this field is 0x001.  If AArch64-TRCIDR0.TRCCCI == 0b0 then this field is zero.  <b>0b000000000100</b>	

## Access

MRS <Xt>, TRCIDR3

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR3	0b10	0b001	0b0000	0b1011	0b111

## Accessibility

MRS <Xt>, TRCIDR3

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR3;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR3;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR3;

```

## A.8.5 TRCIDR4, ID Register 4

Returns the tracing capabilities of the Trace unit.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

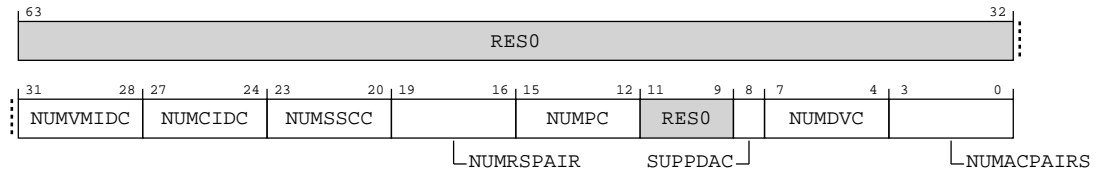
Trace

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-99: AArch64\_trcidr4 bit assignments**



**Table A-272: TRCIDR4 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. <b>0b0001</b> The implementation has one Virtual Context Identifier Comparator.	
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. <b>0b0001</b> The implementation has one Context Identifier Comparator.	
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. <b>0b0001</b> The implementation has one Single-shot Comparator Control.	
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. <b>0b0111</b> The implementation has eight resource selector pairs.	
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. <b>0b0000</b> No PE Comparator Inputs are available.	
[11:9]	RES0	Reserved	0b000
[8]	SUPPDAC	Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other Trace architectures. Allocated in other Trace architectures. <b>0b0</b> Data address comparisons not implemented.	
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other Trace architectures. Allocated in other Trace architectures. <b>0b0000</b> No data value comparators implemented.	
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. <b>0b0100</b> The implementation has four Address Comparator pairs.	

## Access

MRS <Xt>, TRCIDR4

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR4	0b10	0b001	0b0000	0b1100	0b111

## Accessibility

MRS <Xt>, TRCIDR4

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR4;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR4;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR4;

```

## A.8.6 TRCIDR5, ID Register 5

Returns the tracing capabilities of the Trace unit.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

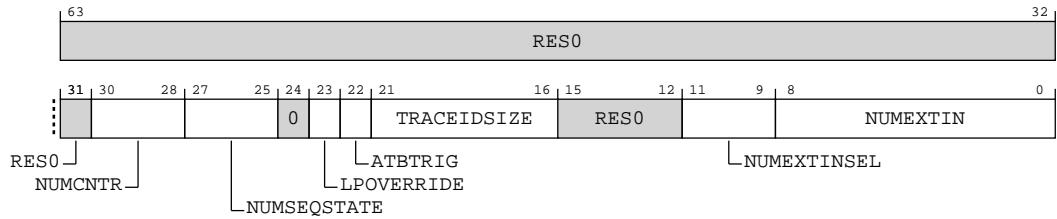
Trace

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-100: AArch64\_trcidr5 bit assignments**



**Table A-274: TRCIDR5 bit descriptions**

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	0x0
[30:28]	NUMCNTR	Indicates the number of Counters that are available for tracing. <b>0b010</b> Two Counters implemented.	
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. <b>0b100</b> Four Sequencer states are implemented.	
[24]	RES0	Reserved	0b0
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. <b>0b0</b> The Trace unit does not support Low-power Override Mode.	
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. <b>0b1</b> The implementation supports ATB triggers.	
[21:16]	TRACEIDSIZE	Indicates the Trace ID width. <b>0b000111</b> The implementation supports a 7-bit Trace ID.	
[15:12]	RES0	Reserved	0b0000
[11:9]	NUMEXTINSEL	Indicates how many External Input Selector resources are implemented. <b>0b100</b> 4 External Input Selector resources are available.	
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. <b>0b11111111</b> Unified PMU event selection.	

## Access

MRS <Xt>, TRCIDR5

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR5	0b10	0b001	0b0000	0b1101	0b111

## Accessibility

MRS <Xt>, TRCIDR5

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR5;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR5;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR5;

```

## A.8.7 TRCIDR10, ID Register 10

Returns the tracing capabilities of the Trace unit.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

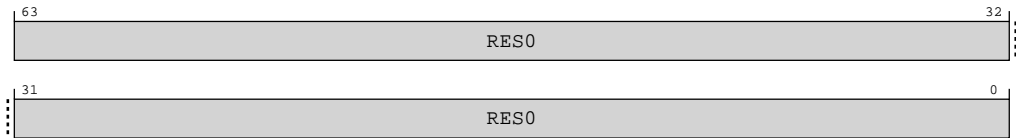
Trace

#### Reset value

0x0

## Bit descriptions

**Figure A-101: AArch64\_trcidr10 bit assignments**



**Table A-276: TRCIDR10 bit descriptions**

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

### Access

MRS <Xt>, TRCIDR10

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR10	0b10	0b001	0b0000	0b0010	0b110

### Accessibility

MRS <Xt>, TRCIDR10

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;

```



### A.8.8 TRCIDR11, ID Register 11

Returns the tracing capabilities of the Trace unit.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

Trace

##### Reset value

0x0

#### Bit descriptions

Figure A-102: AArch64\_trcidr11 bit assignments

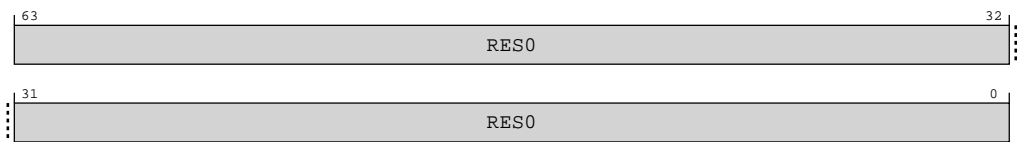


Table A-278: TRCIDR11 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

#### Access

MRS <Xt>, TRCIDR11

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR11	0b10	0b001	0b0000	0b0011	0b110

#### Accessibility

MRS <Xt>, TRCIDR11

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
```

```
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR11;
    elsif PSTATE.EL == EL2 then
        if CPTR_EL2.TTA == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return TRCIDR11;
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return TRCIDR11;
```

A.8.9 TRCIDR12, ID Register 12

Returns the tracing capabilities of the Trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

0x0

Bit descriptions

Figure A-103: AArch64\_trcidr12 bit assignments

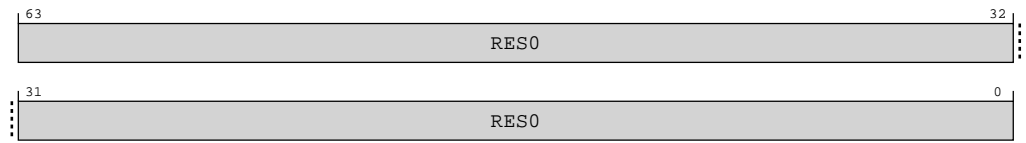


Table A-280: TRCIDR12 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

Access

MRS <Xt>, TRCIDR12

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR12	0b10	0b001	0b0000	0b0100	0b110

## Accessibility

MRS <Xt>, TRCIDR12

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR12;
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR12;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR12;

```

## A.8.10 TRCIDR13, ID Register 13

Returns the tracing capabilities of the Trace unit.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

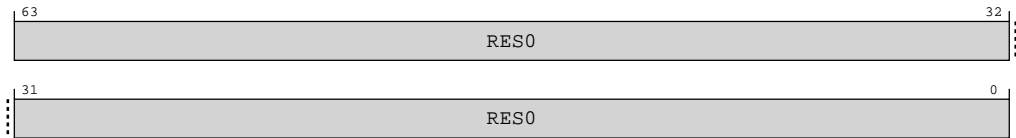
Trace

#### Reset value

0x0

## Bit descriptions

**Figure A-104: AArch64\_trcidr13 bit assignments**



**Table A-282: TRCIDR13 bit descriptions**

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

## Access

MRS <Xt>, TRCIDR13

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR13	0b10	0b001	0b0000	0b0101	0b110

## Accessibility

MRS <Xt>, TRCIDR13

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;

```

## A.8.11 TRCIDR0, ID Register 0

Returns the tracing capabilities of the Trace unit.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

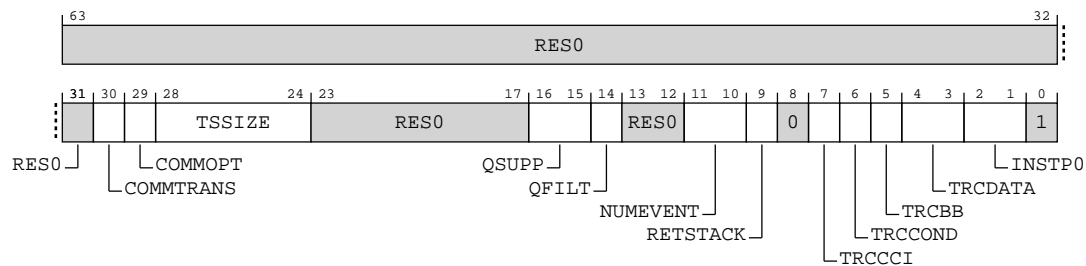
Trace

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-105: AArch64\_trcidr0 bit assignments**



**Table A-284: TRCIDR0 bit descriptions**

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	0x0
[30]	COMMTRANS	Transaction Start element behavior. <b>0b0</b> Transaction Start elements are P0 elements.	
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. <b>0b1</b> Commit mode 1.	
[28:24]	TSSIZE	Indicates that the Trace unit implements Global timestamping and the size of the timestamp value. <b>0b01000</b> Global timestamping implemented with a 64-bit timestamp value.	
[23:17]	RES0	Reserved	0b0000000
[16:15]	QSUPP	Indicates that the Trace unit implements Q element support. <b>0b00</b> Q element support is not implemented.	

Bits	Name	Description	Reset
[14]	QFILT	Indicates if the Trace unit implements Q element filtering. <b>0b0</b> Q element filtering is not implemented.	
[13:12]	RES0	Reserved	0b00
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. <b>0b11</b> The Trace unit supports 4 ETEEvents.	
[9]	RETSTACK	Indicates if the Trace unit supports the return stack. <b>0b1</b> Return stack implemented.	
[8]	RES0	Reserved	0b0
[7]	TRCCCI	Indicates if the Trace unit implements cycle counting. <b>0b1</b> Cycle counting implemented.	
[6]	TRCCOND	Indicates if the Trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other Trace architectures. <b>0b0</b> Conditional instruction tracing not implemented.	
[5]	TRCBB	Indicates if the Trace unit implements branch broadcasting. <b>0b1</b> Branch broadcasting implemented.	
[4:3]	TRCDATA	Indicates if the Trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other Trace architectures. <b>0b00</b> Tracing of data addresses and data values is not implemented.	
[2:1]	INSTPO	Indicates if load and store instructions are PO instructions. Load and store instructions as PO instructions is not implemented in ETE and this field is reserved for other Trace architectures. <b>0b00</b> Load and store instructions are not PO instructions.	
[0]	RES1	Reserved	0b1

## Access

MRS <Xt>, TRCIDRO

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDRO	0b10	0b001	0b0000	0b1000	0b111

## Accessibility

MRS <Xt>, TRCIDRO

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then

```

```
    AArch64.SystemAccessTrap(EL1, 0x18);
elseif EL2Enabled() && CPTR_EL2.TTA == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elseif CPTR_EL3.TTA == '1' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    return TRCIDR0;
elseif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR0;
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR0;
```

A.8.12 TRCIDR1, ID Register 1

Returns the tracing capabilities of the Trace unit.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

See individual bit resets.

Bit descriptions

Figure A-106: AArch64\_trcidr1 bit assignments

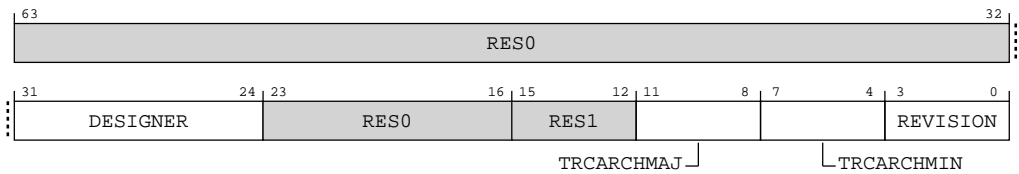


Table A-286: TRCIDR1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[31:24]	DESIGNER	Indicates which company designed the Trace unit. The permitted values of this field are the same as AArch64-MIDR_EL1.Implementer.  <b>0b01000001</b> Arm Limited	
[23:16]	RES0	Reserved	0b00000000
[15:12]	RES1	Reserved	0b1111
[11:8]	TRCARCHMAJ	Major architecture version.  <b>0b1111</b> If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to AArch64-TRCDEVARCH.	
[7:4]	TRCARCHMIN	Minor architecture version.  <b>0b1111</b> If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to AArch64-TRCDEVARCH.	
[3:0]	REVISION	Implementation revision that identifies the revision of the Trace and OS Lock registers.  <b>0b0010</b> Revision 2	

## Access

MRS <Xt>, TRCIDR1

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR1	0b10	0b001	0b0000	0b1001	0b111

## Accessibility

MRS <Xt>, TRCIDR1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HDFGRTR_EL2.TRCID == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;
    end
elsif PSTATE.EL == EL2 then
    if CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;
    end
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;
    end
end

```



### A.8.13 TRCCIDCVR0, Context Identifier Comparator Value Registers <n>

Contains a Context identifier value.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

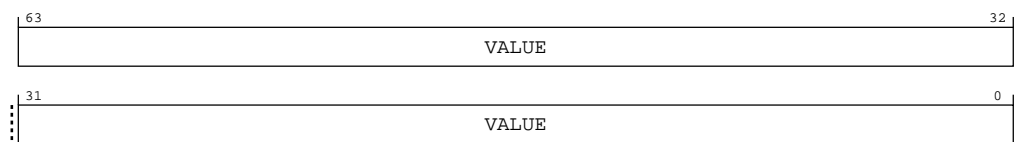
Trace

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure A-107: AArch64\_trccidcvr0 bit assignments**



**Table A-288: TRCCIDCVR0 bit descriptions**

Bits	Name	Description	Reset
[63:0]	VALUE	Context identifier value. The width of this field is indicated by AArch64-TRCIDR2.CIDSIZE. Unimplemented bits are RES0. After a PE Reset, the Trace unit assumes that the Context identifier is zero until the PE updates the Context identifier.	

#### Access

MRS <Xt>, TRCCIDCVR0

<systemreg>	op0	op1	CRn	CRm	op2
TRCCIDCVR0	0b10	0b001	0b0011	0b0000	0b000

MSR TRCCIDCVR0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
TRCCIDCVR0	0b10	0b001	0b0011	0b0000	0b000

## A.9 MPAM register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** MPAM registers in the core. Individual register descriptions provide detailed information.

**Table A-291: MPAM register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
<a href="#">MPAMVPMV_EL2</a>	3	C10	4	C4	1	See individual bit resets.	64-bit	MPAM Virtual Partition Mapping Valid Register
<a href="#">MPAMVPM0_EL2</a>	3	C10	4	C6	0	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 0
<a href="#">MPAMVPM1_EL2</a>	3	C10	4	C6	1	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 1
<a href="#">MPAMVPM7_EL2</a>	3	C10	4	C6	7	See individual bit resets.	64-bit	MPAM Virtual PARTID Mapping Register 7

### A.9.1 MPAMVPMV\_EL2, MPAM Virtual Partition Mapping Valid Register

Valid bits for virtual PARTID mapping entries. Each bit *m* corresponds to virtual PARTID mapping entry *m* in the MPAMVPM<*n*>\_EL2 registers where *n* = *m* >> 2.

#### Configurations

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

#### Attributes

##### Width

64

##### Functional group

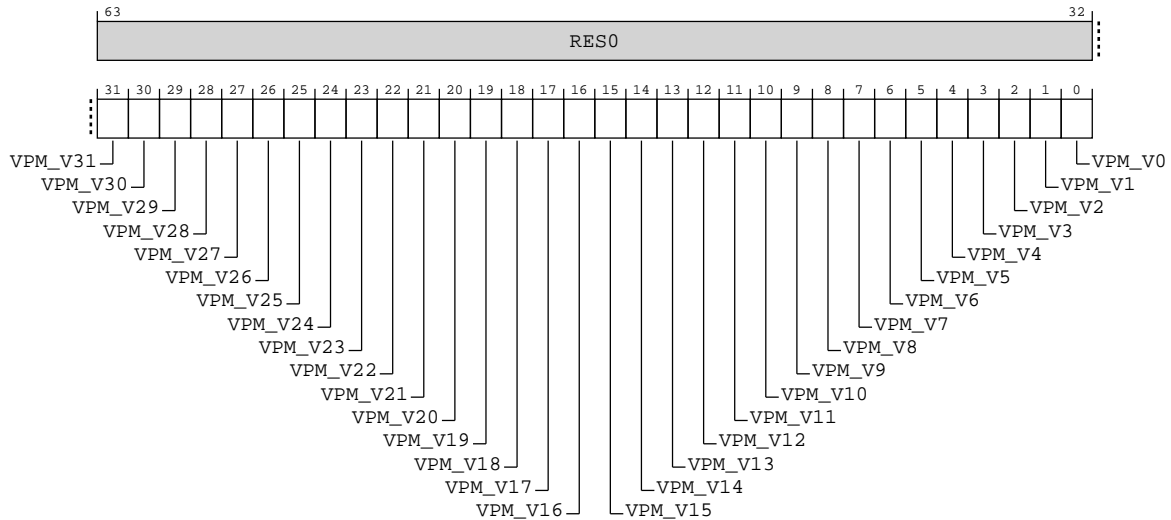
MPAM

##### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-108: AArch64\_mpamvpmv\_el2 bit assignments**



**Table A-292: MPAMVPMV\_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31]	VPM_V31	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[30]	VPM_V30	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[29]	VPM_V29	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[28]	VPM_V28	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[27]	VPM_V27	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[26]	VPM_V26	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[25]	VPM_V25	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[24]	VPM_V24	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[23]	VPM_V23	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[22]	VPM_V22	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[21]	VPM_V21	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[20]	VPM_V20	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[19]	VPM_V19	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[18]	VPM_V18	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[17]	VPM_V17	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[16]	VPM_V16	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[15]	VPM_V15	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[14]	VPM_V14	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[13]	VPM_V13	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[12]	VPM_V12	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[11]	VPM_V11	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	

Bits	Name	Description	Reset
[10]	VPM_V10	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[9]	VPM_V9	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[8]	VPM_V8	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[7]	VPM_V7	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[6]	VPM_V6	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[5]	VPM_V5	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[4]	VPM_V4	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[3]	VPM_V3	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[2]	VPM_V2	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[1]	VPM_V1	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	
[0]	VPM_V0	Contains valid bit for virtual PARTID mapping entry corresponding to virtual PARTID<m>.	

## Access

MRS <Xt>, MPAMVPMV\_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPMV_EL2	0b11	0b100	0b1010	0b0100	0b001

MSR MPAMVPMV\_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPMV_EL2	0b11	0b100	0b1010	0b0100	0b001

## Accessibility

MRS <Xt>, MPAMVPMV\_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x938];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elsif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPMV_EL2;
    elsif PSTATE.EL == EL3 then
        return MPAMVPMV_EL2;

```

MSR MPAMVPMV\_EL2, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;

```

```

elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x938] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            MPAMVPMV_EL2 = X[t];
    elseif PSTATE.EL == EL3 then
        MPAMVPMV_EL2 = X[t];

```

## A.9.2 MPAMVPM0\_EL2, MPAM Virtual PARTID Mapping Register 0

MPAMVPM0\_EL2 provides mappings from virtual PARTIDs 0 - 3 to physical PARTIDs.

AArch64-MPAMIDR\_EL1.VPMR\_MAX field gives the index of the highest implemented MPAMVPM<n>\_EL2 register. VPMR\_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR\_EL1.VPMR\_MAX == 0, there is only a single MPAMVPM<n>\_EL2 register, AArch64-MPAMVPM0\_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR\_EL2.EL1\_VPMEN for PARTIDs in AArch64-MPAM1\_EL1 and by AArch64-MPAMHCR\_EL2.EL0\_VPMEN for PARTIDs in AArch64-MPAM0\_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV\_EL2.VPM\_V bit in bit position n is set to 1.

### Configurations

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

### Attributes

#### Width

64

#### Functional group

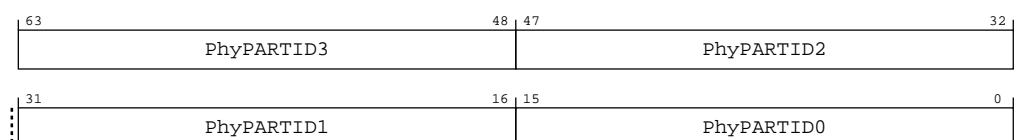
MPAM

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-109: AArch64\_mpamvpm0\_el2 bit assignments**



**Table A-295: MPAMVPMO\_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:48]	PhyPARTID3	Virtual PARTID Mapping Entry for virtual PARTID 3. PhyPARTID3 gives the mapping of virtual PARTID 3 to a physical PARTID.	
[47:32]	PhyPARTID2	Virtual PARTID Mapping Entry for virtual PARTID 2. PhyPARTID2 gives the mapping of virtual PARTID 2 to a physical PARTID.	
[31:16]	PhyPARTID1	Virtual PARTID Mapping Entry for virtual PARTID 1. PhyPARTID1 gives the mapping of virtual PARTID 1 to a physical PARTID.	
[15:0]	PhyPARTID0	Virtual PARTID Mapping Entry for virtual PARTID 0. PhyPARTID0 gives the mapping of virtual PARTID 0 to a physical PARTID.	

### Access

MRS &lt;Xt&gt;, MPAMVPMO\_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPMO_EL2	0b11	0b100	0b1010	0b0110	0b000

MSR MPAMVPMO\_EL2, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPMO_EL2	0b11	0b100	0b1010	0b0110	0b000

### Accessibility

MRS &lt;Xt&gt;, MPAMVPMO\_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x940];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elsif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPMO_EL2;
    elsif PSTATE.EL == EL3 then
        return MPAMVPMO_EL2;

```

MSR MPAMVPMO\_EL2, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x940] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then

```

```
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM0_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    MPAMVPM0_EL2 = X[t];
```

A.9.3 MPAMVPM1\_EL2, MPAM Virtual PARTID Mapping Register 1

MPAMVPM1\_EL2 provides mappings from virtual PARTIDs 4 - 7 to physical PARTIDs.

AArch64-MPAMIDR\_EL1.VPMR\_MAX field gives the index of the highest implemented AArch64-MPAMVPM0\_EL2 to AArch64-MPAMVPM7\_EL2 registers. VPMR\_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR\_EL1.VPMR\_MAX == 0, there is only a single MPAMVPM<n>\_EL2 register, AArch64-MPAMVPM0\_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR\_EL2.EL1\_VPMEN for PARTIDs in AArch64-MPAM1\_EL1 and by MPAMHCR\_EL2.ELO\_VPMEN for PARTIDs in AArch64-MPAM0\_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV\_EL2.VPM\_V bit in bit position n is set to 1.

Configurations

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

Attributes

Width

64

Functional group

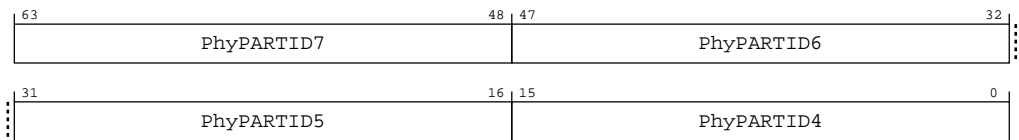
MPAM

Reset value

See individual bit resets.

Bit descriptions

Figure A-110: AArch64\_mpamvpm1\_el2 bit assignments



**Table A-298: MPAMVPM1\_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:48]	PhyPARTID7	Virtual PARTID Mapping Entry for virtual PARTID 7. PhyPARTID7 gives the mapping of virtual PARTID 7 to a physical PARTID.	
[47:32]	PhyPARTID6	Virtual PARTID Mapping Entry for virtual PARTID 6. PhyPARTID6 gives the mapping of virtual PARTID 6 to a physical PARTID.	
[31:16]	PhyPARTID5	Virtual PARTID Mapping Entry for virtual PARTID 5. PhyPARTID5 gives the mapping of virtual PARTID 5 to a physical PARTID.	
[15:0]	PhyPARTID4	Virtual PARTID Mapping Entry for virtual PARTID 4. PhyPARTID4 gives the mapping of virtual PARTID 4 to a physical PARTID.	

### Access

MRS &lt;Xt&gt;, MPAMVPM1\_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM1_EL2	0b11	0b100	0b1010	0b0110	0b001

MSR MPAMVPM1\_EL2, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM1_EL2	0b11	0b100	0b1010	0b0110	0b001

### Accessibility

MRS &lt;Xt&gt;, MPAMVPM1\_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x948];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elseif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPM1_EL2;
    elseif PSTATE.EL == EL3 then
        return MPAMVPM1_EL2;

```

MSR MPAMVPM1\_EL2, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x948] = X[t];
    elseif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then

```



```
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM1_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    MPAMVPM1_EL2 = X[t];
```

### A.9.4 MPAMVPM7\_EL2, MPAM Virtual PARTID Mapping Register 7

MPAMVPM7\_EL2 provides mappings from virtual PARTIDs 28 - 31 to physical PARTIDs.

AArch64-MPAMIDR\_EL1.VPMR\_MAX field gives the index of the highest implemented MPAMVPM<n>\_EL2 registers. VPMR\_MAX can be as large as 7 (8 registers) or 32 virtual PARTIDs. If AArch64-MPAMIDR\_EL1.VPMR\_MAX == 0, there is only a single MPAMVPM<n>\_EL2 register, AArch64-MPAMVPM0\_EL2. Virtual PARTID mapping is enabled by AArch64-MPAMHCR\_EL2.EL1\_VPMEN for PARTIDs in AArch64-MPAM1\_EL1 and by AArch64-MPAMHCR\_EL2.ELO\_VPMEN for AArch64-MPAM0\_EL1. A virtual-to-physical PARTID mapping entry, PhyPARTID<n>, is only valid when the AArch64-MPAMVPMV\_EL2.VPM\_V bit in bit position n is set to 1.

#### Configurations

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

#### Attributes

##### Width

64

##### Functional group

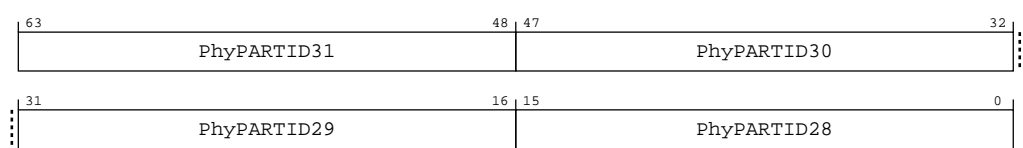
MPAM

##### Reset value

See individual bit resets.

#### Bit descriptions

Figure A-111: AArch64\_mpamvpm7\_el2 bit assignments



**Table A-301: MPAMVPM7\_EL2 bit descriptions**

Bits	Name	Description	Reset
[63:48]	PhyPARTID31	Virtual PARTID Mapping Entry for virtual PARTID 31. PhyPARTID31 gives the mapping of virtual PARTID 31 to a physical PARTID.	
[47:32]	PhyPARTID30	Virtual PARTID Mapping Entry for virtual PARTID 30. PhyPARTID30 gives the mapping of virtual PARTID 30 to a physical PARTID.	
[31:16]	PhyPARTID29	Virtual PARTID Mapping Entry for virtual PARTID 29. PhyPARTID29 gives the mapping of virtual PARTID 29 to a physical PARTID.	
[15:0]	PhyPARTID28	Virtual PARTID Mapping Entry for virtual PARTID 28. PhyPARTID28 gives the mapping of virtual PARTID 28 to a physical PARTID.	

### Access

MRS &lt;Xt&gt;, MPAMVPM7\_EL2

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM7_EL2	0b11	0b100	0b1010	0b0110	0b111

MSR MPAMVPM7\_EL2, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
MPAMVPM7_EL2	0b11	0b100	0b1010	0b0110	0b111

### Accessibility

MRS &lt;Xt&gt;, MPAMVPM7\_EL2

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        return NVMem[0x978];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            UNDEFINED;
    elsif PSTATE.EL == EL2 then
        if MPAM3_EL3.TRAPLOWER == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return MPAMVPM7_EL2;
    elsif PSTATE.EL == EL3 then
        return MPAMVPM7_EL2;

```

MSR MPAMVPM7\_EL2, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.<NV2,NV> == '11' then
        NVMem[0x978] = X[t];
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        if MPAM3_EL3.TRAPLOWER == '1' then

```

```

        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elseif PSTATE.EL == EL2 then
    if MPAM3_EL3.TRAPLOWER == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MPAMVPM7_EL2 = X[t];
elseif PSTATE.EL == EL3 then
    MPAMVPM7_EL2 = X[t];

```

## A.10 AArch64 RAS register summary

The summary table provides an overview of all **IMPLEMENTATION DEFINED** RAS registers in the core. Individual register descriptions provide detailed information.

**Table A-304: RAS register summary**

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
<a href="#">ERRIDR_EL1</a>	3	C5	0	C3	0	See individual bit resets.	64-bit	Error Record ID Register
<a href="#">ERRSELR_EL1</a>	3	C5	0	C3	1	See individual bit resets.	64-bit	Error Record Select Register
<a href="#">ERXFR_EL1</a>	3	C5	0	C4	0	See individual bit resets.	64-bit	Selected Error Record Feature Register
<a href="#">ERXCTLR_EL1</a>	3	C5	0	C4	1	0x0	64-bit	Selected Error Record Control Register
<a href="#">ERXSTATUS_EL1</a>	3	C5	0	C4	2	0x0	64-bit	Selected Error Record Primary Status Register
<a href="#">ERXADDR_EL1</a>	3	C5	0	C4	3	See individual bit resets.	64-bit	Selected Error Record Address Register
<a href="#">ERXPFGF_EL1</a>	3	C5	0	C4	4	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Feature register
<a href="#">ERXPFGCTL_EL1</a>	3	C5	0	C4	5	0x0	64-bit	Selected Pseudo-fault Generation Control register
<a href="#">ERXPFGCDN_EL1</a>	3	C5	0	C4	6	See individual bit resets.	64-bit	Selected Pseudo-fault Generation Countdown register
<a href="#">ERXMISCO_EL1</a>	3	C5	0	C5	0	See individual bit resets.	64-bit	Selected Error Record Miscellaneous Register 0
<a href="#">ERXMISC1_EL1</a>	3	C5	0	C5	1	0x0	64-bit	Selected Error Record Miscellaneous Register 1
<a href="#">ERXMISC2_EL1</a>	3	C5	0	C5	2	0x0	64-bit	Selected Error Record Miscellaneous Register 2
<a href="#">ERXMISC3_EL1</a>	3	C5	0	C5	3	0x0	64-bit	Selected Error Record Miscellaneous Register 3

### A.10.1 ERRIDR\_EL1, Error Record ID Register

Defines the highest numbered index of the error records that can be accessed through the Error Record System registers.

#### Configurations

This register is available in all configurations.

#### Attributes

##### Width

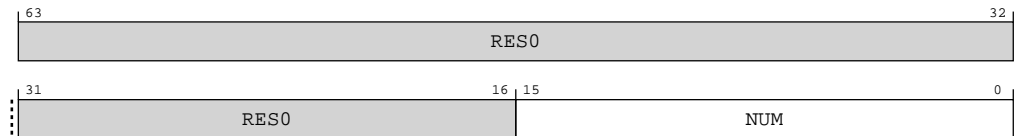
64

**Functional group**

RAS

**Reset value**

See individual bit resets.

**Bit descriptions****Figure A-112: AArch64\_erridr\_el1 bit assignments****Table A-305: ERRIDR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0x0
[15:0]	NUM	<p>Highest numbered index of the records that can be accessed through the Error Record System registers plus one. Zero indicates no records can be accessed through the Error Record System registers.</p> <p>Each implemented record is owned by a node. A node might own multiple records.</p> <p><b>0b00000000000000010</b></p> <p>Two Records Present.</p>	

**Access**

MRS &lt;Xt&gt;, ERRIDR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERRIDR_EL1	0b11	0b000	0b0101	0b0011	0b000

**Accessibility**

MRS &lt;Xt&gt;, ERRIDR\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERRIDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRIDR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRIDR_EL1;
elsif PSTATE.EL == EL3 then

```

```
return ERRIDR_EL1;
```

### A.10.2 ERRSELR\_EL1, Error Record Select Register

Selects an error record to be accessed through the Error Record System registers.

#### Configurations

If AArch64-ERRIDR\_EL1 indicates that zero error records are implemented, then it is IMPLEMENTATION DEFINED whether ERRSELR\_EL1 is UNDEFINED or RES0.

#### Attributes

##### Width

64

##### Functional group

RAS

##### Reset value

See individual bit resets.

#### Bit descriptions

Figure A-113: AArch64\_errselr\_el1 bit assignments

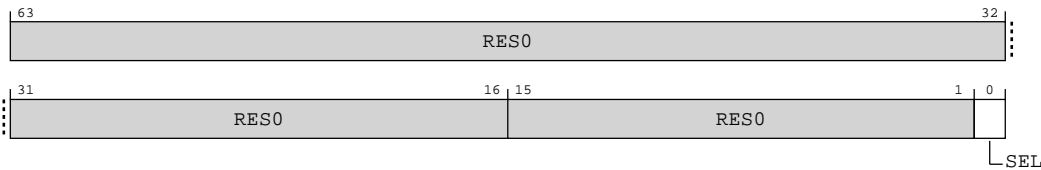


Table A-307: ERRSELR\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:1]	RES0	Reserved	0x0
[0]	SEL	<b>0b0</b> Selects record 0, containing errors from DSU RAMs <b>0b1</b> Selects record 1, containing errors from Core RAMs	

#### Access

MRS <Xt>, ERRSELR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERRSELR_EL1	0b11	0b000	0b0101	0b0011	0b001

MSR ERRSELR\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERRSELR_EL1	0b11	0b000	0b0101	0b0011	0b001

## Accessibility

MRS <Xt>, ERRSELR\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERRSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRSELR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERRSELR_EL1;
elsif PSTATE.EL == EL3 then
    return ERRSELR_EL1;

```

MSR ERRSELR\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERRSELR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERRSELR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERRSELR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERRSELR_EL1 = X[t];

```

## A.10.3 ERXFR\_EL1, Selected Error Record Feature Register

Accesses ext-ERR<n>FR for the error record <n> selected by AArch64-ERRSELR\_EL1.SEL.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

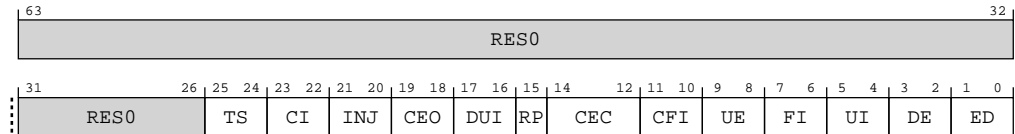
64

**Functional group**

RAS

**Reset value**

See individual bit resets.

**Bit descriptions****Figure A-114: AArch64\_erxfr\_el1 bit assignments****Table A-310: ERXFR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:26]	RES0	Reserved	0b000000
[25:24]	TS	Timestamp Extension. Indicates whether, for each error record <m> owned by this node, rERXMISC3_EL1 is used as the timestamp register, and, if it is, the timebase used by the timestamp. <b>0b00</b> The node does not support a timestamp register.	
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented. <b>0b00</b> Does not support the critical error interrupt. ERXCTLR_EL1.CI is RES0.	
[21:20]	INJ	Fault Injection Extension. Indicates whether the RAS Common Fault Injection Model Extension is implemented. <b>0b01</b> The node implements the RAS Common Fault Injection Model Extension. See ERXPFGF_EL1 for more information.	
[19:18]	CEO	Corrected Error overwrite. Indicates the behavior when a second Corrected error is detected after a first Corrected error has been recorded by an error record <m> owned by the node. <b>0b00</b> Counts Corrected errors if a counter is implemented. Keeps the previous error syndrome. If the counter overflows, or no counter is implemented, then ERXSTATUS_EL1.OF is set to 0b1.	
[17:16]	DUI	Error recovery interrupt for deferred errors control. Indicates whether the control for enabling error recovery interrupts on deferred errors are implemented. <b>0b00</b> Does not support the control for enabling error recovery interrupts on deferred errors. ERXCTLR_EL1.DUI is RES0.	
[15]	RP	Repeat counter. Indicates whether the node implements the repeat Corrected error counter in ERXMISC0_EL1 for each error record <m> owned by the node that implements the standard Corrected error counter. <b>0b1</b> A first (repeat) counter and a second (other) counter are implemented. The repeat counter is the same size as the primary error counter.	

Bits	Name	Description	Reset
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter (CE counter) mechanisms in ERXMISCO_EL1 for each error record <m> owned by the node that can record countable errors.  <b>0b010</b> Implements an 8-bit Corrected error counter in ERXMISCO_EL1[39:32].	
[11:10]	CFI	Fault handling interrupt for corrected errors. Indicates whether the control for enabling fault handling interrupts on corrected errors are implemented.  <b>0b10</b> Control for enabling fault handling interrupts on corrected errors is supported and controllable using ERXCTLR_EL1.CFI.	
[9:8]	UE	In-band uncorrected error reporting. Indicates whether the in-band uncorrected error reporting (External Aborts) and associated controls are implemented.  <b>0b01</b> In-band uncorrected error reporting (External Aborts) is supported and always enabled. ERXCTLR_EL1.UE is RES0.	
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented.  <b>0b10</b> Fault handling interrupt is supported and controllable using ERXCTLR_EL1.FI.	
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented.  <b>0b10</b> Error handling interrupt is supported and controllable using ERXCTLR_EL1.UI.	
[3:2]	DE	<b>0b00</b>	
[1:0]	ED	Error reporting and logging. Indicates whether error record <n> is the first record owned the node, and, if so, whether it implements the controls for enabling and disabling error reporting and logging.  <b>0b10</b> Error reporting and logging is controllable using ERXCTLR_EL1.ED.	

## Access

MRS <Xt>, ERXFR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXFR_EL1	0b11	0b000	0b0101	0b0100	0b000

## Accessibility

MRS <Xt>, ERXFR\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXFR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else

```



```
        return ERXFR_EL1;
    elsif PSTATE.EL == EL2 then
        if SCR_EL3.TERR == '1' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return ERXFR_EL1;
    elsif PSTATE.EL == EL3 then
        return ERXFR_EL1;
```

A.10.4 ERXCTLR\_EL1, Selected Error Record Control Register

Accesses ext-ERR<n>CTLR for the error record <n> selected by AArch64-ERRSELR\_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

0x0

Bit descriptions

Figure A-115: AArch64\_erxctlr\_el1 bit assignments

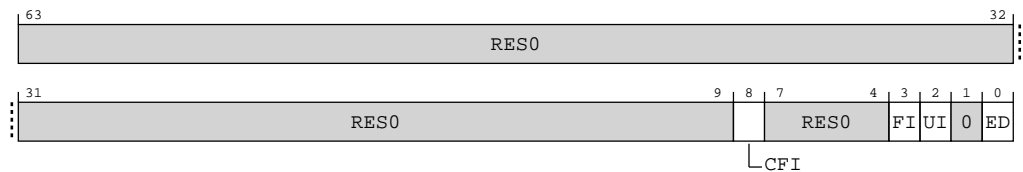


Table A-312: ERXCTLR\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:9]	RES0	Reserved	0x0

Bits	Name	Description	Reset
[8]	CFI	<p>Fault handling interrupt for Corrected errors enable.</p> <p>This control applies to errors arising from both reads and writes.</p> <p>The fault handling interrupt is generated when one of the standard CE counters on ERXMISCO_EL1 overflows and the overflow bit is set. The possible values are:</p> <p><b>0b0</b></p> <p>Fault handling interrupt not generated for Corrected errors.</p> <p><b>0b1</b></p> <p>Fault handling interrupt generated for Corrected errors.</p> <p>The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0x0
[7:4]	RES0	Reserved	0b0000
[3]	FI	<p>Fault handling interrupt enable.</p> <p>This control applies to errors arising from both reads and writes.</p> <p>The fault handling interrupt is generated for all detected Deferred errors and Uncorrected errors. The possible values are:</p> <p><b>0b0</b></p> <p>Fault handling interrupt disabled.</p> <p><b>0b1</b></p> <p>Fault handling interrupt enabled.</p> <p>The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0b0
[2]	UI	<p>Uncorrected error recovery interrupt enable.</p> <p>This control applies to errors arising from both reads and writes.</p> <p>When enabled, the error recovery interrupt is generated for all detected Uncorrected errors that are not deferred.</p> <p><b>0b0</b></p> <p>Error recovery interrupt disabled.</p> <p><b>0b1</b></p> <p>Error recovery interrupt enabled.</p> <p>The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0b0
[1]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[0]	ED	Error Detection and correction enable. The possible values are:  <b>0b0</b> Error detection and correction disabled.  <b>0b1</b> Error detection and correction enabled.  Cold reset only. Unaffected by Warm reset	0b0

## Access

MRS <Xt>, ERXCTLR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXCTLR_EL1	0b11	0b000	0b0101	0b0100	0b001

MSR ERXCTLR\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXCTLR_EL1	0b11	0b000	0b0101	0b0100	0b001

## Accessibility

MRS <Xt>, ERXCTLR\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXCTLR_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXCTLR_EL1;
elsif PSTATE.EL == EL3 then
    return ERXCTLR_EL1;

```

MSR ERXCTLR\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXCTLR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXCTLR_EL1 = X[t];

```

```
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXCTLR_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ERXCTLR_EL1 = X[t];
```

A.10.5 ERXSTATUS\_EL1, Selected Error Record Primary Status Register

Accesses ext-ERR<n>STATUS for the error record <n> selected by AArch64-ERRSELR\_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

0x0

Bit descriptions

Figure A-116: AArch64\_erxstatus\_el1 bit assignments

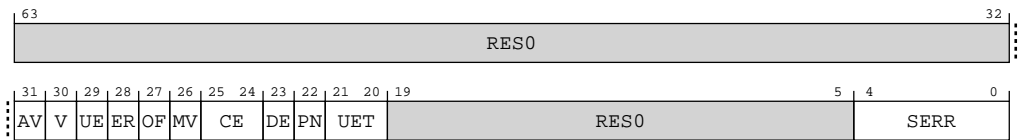


Table A-315: ERXSTATUS\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31]	AV	<p>Address Valid. The possible values are:</p> <p><b>0b0</b></p> <p>ERXADDR_EL1 not valid.</p> <p><b>0b1</b></p> <p>ERXADDR_EL1 contains an address associated with the highest priority error recorded by this record.</p> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0x0

Bits	Name	Description	Reset
[30]	V	<p>Status Register Valid. The possible values are:</p> <p><b>0b0</b> ERXSTATUS_EL1 not valid.</p> <p><b>0b1</b> ERXSTATUS_EL1 valid. At least one error has been recorded.</p> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0x0
[29]	UE	<p>Uncorrected Error. The possible values are:</p> <p><b>0b0</b> No errors have been detected, or all detected errors have been either corrected or deferred.</p> <p><b>0b1</b> At least one detected error was not corrected and not deferred.</p> <p>When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero.</p> <p>This bit is not valid and reads UNKNOWN if ERXSTATUS_EL1.V == 0b0.</p> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0x0
[28]	ER	<p>Error Reported. The possible values are:</p> <p><b>0b0</b> No in-band error (External Abort) reported.</p> <p><b>0b1</b> An External Abort was signaled by the node to the master making the access or other transaction.</p> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p> <p><b>Note:</b> An External Abort signaled by the node might be masked and not generate any exception.</p>	0x0

Bits	Name	Description	Reset
[27]	OF	<p>Overflow. The possible values are:</p> <p><b>0b0</b></p> <p>If UE == 1, then no error status for an Uncorrected error has been discarded.</p> <p>If UE == 0 and DE == 1, then no error status for a Deferred error has been discarded.</p> <p>If UE == 0, DE == 0, and CE != 0b00, then the corrected error counter has not overflowed.</p> <p><b>0b1</b></p> <p>More than one error has occurred and so details of the other error have been discarded.</p> <p>When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero.</p> <p>This bit is not valid and reads UNKNOWN if ERXSTATUS_EL1.V == 0b0.</p> <p>Cold reset only. Unaffected by Warm reset</p> <p>This bit is read/write-one-to-clear.</p>	0x0
[26]	MV	<p>Miscellaneous Registers Valid. The possible values are:</p> <p><b>0b0</b></p> <p>ERXMISC&lt;m&gt;_EL1 not valid.</p> <p><b>0b1</b></p> <p>This bit indicates that the ERXMISC&lt;m&gt;_EL1 registers contain additional information for an error recorded by this record.</p> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p> <p><b>Note:</b></p> <p>If the ERXMISC&lt;m&gt;_EL1 registers can contain additional information for a previously recorded error, then the contents must be self-describing to software or a user. For example, certain fields might relate only to Corrected errors, and other fields only to the most recent error that was not discarded.</p>	0x0

Bits	Name	Description	Reset
[25:24]	CE	<p>Corrected Error. The possible values are:</p> <p><b>0b00</b> No errors were corrected.</p> <p><b>0b01</b> At least one transient error was corrected.</p> <p><b>0b10</b> At least one error was corrected.</p> <p><b>0b11</b> At least one persistent error was corrected.</p> <p>When clearing ERXSTATUS_EL1.V to 0b0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero.</p> <p>This field is not valid and reads UNKNOWN if ERXSTATUS_EL1.V == 0b0.</p> <p>This field is read/write-ones-to-clear. Writing a value other than all-zeros or all-ones sets this field to an UNKNOWN value.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0x0
[23]	DE	<p>Deferred Error. The possible values are:</p> <p><b>0b0</b> No errors were deferred.</p> <p><b>0b1</b> At least one error was not corrected and deferred.</p> <p>When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero.</p> <p>This bit is not valid and reads UNKNOWN if ERXSTATUS_EL1.V == 0b0.</p> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0x0
[22]	PN	<p>Poison. The value is:</p> <p><b>0b0</b> This core cannot distinguish a poisoned value from a corrupted value.</p> <p>When clearing ERXSTATUS_EL1.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero.</p> <p>This bit is not valid and reads UNKNOWN if any of the following are true:</p> <ul style="list-style-type: none"> <li>ERXSTATUS_EL1.V == 0b0.</li> <li>ERXSTATUS_EL1.{DE,UE} == {0,0}.</li> </ul> <p>This bit is read/write-one-to-clear.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0x0

Bits	Name	Description	Reset
[21:20]	UET	Uncorrected Error Type. The value is:  <b>0b00</b> Uncorrected error, Uncontainable error (UC).  Cold reset only. Unaffected by Warm reset	0x0
[19:5]	RES0	Reserved	0x0
[4:0]	SERR	Primary error code.  The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry.  The possible values are: <b>0b000000</b> No error <b>0b000010</b> ECC error from internal data buffer. <b>0b000110</b> ECC error on cache data RAM. <b>0b000111</b> ECC error on cache tag or dirty RAM. <b>0b010000</b> Parity error on TLB data RAM. <b>0b100010</b> Error response for a cache copyback. <b>0b101011</b> Deferred error from slave not supported at the consumer. For example, poisoned data received from a slave by a master that cannot defer the error further.  Cold reset only. Unaffected by Warm reset	0x0

## Access

MRS <Xt>, ERXSTATUS\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXSTATUS_EL1	0b11	0b000	0b0101	0b0100	0b010

MSR ERXSTATUS\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXSTATUS_EL1	0b11	0b000	0b0101	0b0100	0b010

## Accessibility

MRS <Xt>, ERXSTATUS\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;

```



```

elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXSTATUS_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXSTATUS_EL1;
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXSTATUS_EL1;
elseif PSTATE.EL == EL3 then
    return ERXSTATUS_EL1;

```

MSR ERXSTATUS\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXSTATUS_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXSTATUS_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXSTATUS_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ERXSTATUS_EL1 = X[t];

```

## A.10.6 ERXADDR\_EL1, Selected Error Record Address Register

Accesses ext-ERR<n>ADDR for the error record <n> selected by AArch64-ERRSELR\_EL1.SEL.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

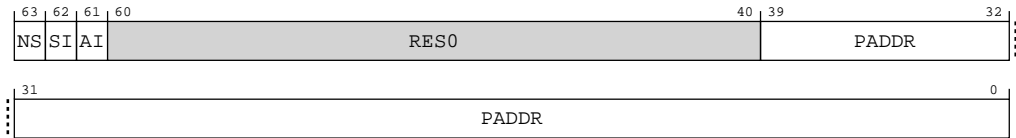
RAS

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-117: AArch64\_erxaddr\_el1 bit assignments**



**Table A-318: ERXADDR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63]	NS	Non-secure attribute. The possible values are:  <b>0b0</b> The address is Secure.  <b>0b1</b> The address is Non-secure.	
[62]	SI	Secure Incorrect. Indicates whether the NS bit is valid. The possible values of this bit are:  <b>0b0</b> The NS bit is correct. That is, it matches the programmers' view of the Non-secure attribute for this recorded location.	
[61]	AI	Address Incomplete or incorrect. Indicates whether the PADDR field is a valid physical address. The possible values of this bit are:  <b>0b0</b> The PADDR field is a valid physical address. That is, it matches the programmers' view of the physical address for this recorded location.	
[60:40]	RES0	Reserved	0x0
[39:0]	PADDR	Physical Address. Address of the recorded location	

## Access

MRS <Xt>, ERXADDR\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXADDR_EL1	0b11	0b000	0b0101	0b0100	0b011

MSR ERXADDR\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXADDR_EL1	0b11	0b000	0b0101	0b0100	0b011

## Accessibility

MRS <Xt>, ERXADDR\_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
```

```

        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXADDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXADDR_EL1;
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXADDR_EL1;
elseif PSTATE.EL == EL3 then
    return ERXADDR_EL1;

```

MSR ERXADDR\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXADDR_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXADDR_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXADDR_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ERXADDR_EL1 = X[t];

```

## A.10.7 ERXPFGF\_EL1, Selected Pseudo-fault Generation Feature register

Accesses ext-ERR<n>PFGF for the error record <n> selected by AArch64-ERRSELR\_EL1.SEL.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

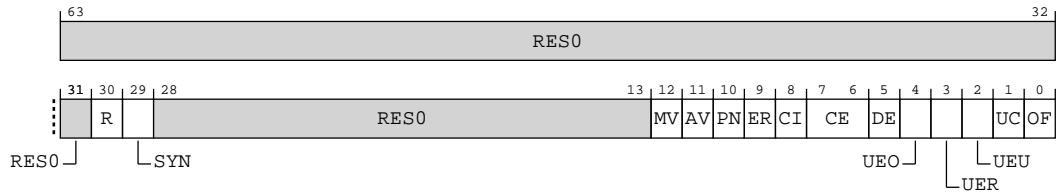
RAS

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure A-118: AArch64\_erpfgf\_el1 bit assignments**



**Table A-321: ERXPFGF\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	0x0
[30]	R	Restartable bit. When it reaches zero, the Error Generation Counter restarts from the ERXPFGCDN_EL1 value or stops. The value is: <b>0b1</b> Feature controllable.	
[29]	SYN	Syndrome. Fault syndrome injection. The value is: <b>0b0</b> When an injected error is recorded, the node sets ERXSTATUS_EL1.{IERR, SERR} to IMPLEMENTATION DEFINED values. ERXSTATUS_EL1.{IERR, SERR} are UNKNOWN when ERXSTATUS_EL1.V == 0b0.	
[28:13]	RES0	Reserved	0x0
[12]	MV	Miscellaneous syndrome.  Additional syndrome injection. Defines whether software can control all or part of the syndrome recorded in the ERXMISC<m>_EL1 registers when an injected error is recorded.  It is <b>IMPLEMENTATION DEFINED</b> which syndrome fields in ERXMISC<m>_EL1 this refers to, as some fields might always be recorded by an error. For example, a Corrected Error counter. <b>0b0</b> When an injected error is recorded, the node might record IMPLEMENTATION DEFINED additional syndrome in ERXMISC<m>_EL1. If any syndrome is recorded in ERXMISC<m>_EL1, then ERXSTATUS_EL1.MV is set to 0b1.  <b>Note:</b> If ERR<n>PFGF.MV == 0b1, software can write specific values into the ERR<n>MISC<m> registers when setting up a fault injection event. The values that can be written to these registers are <b>IMPLEMENTATION DEFINED</b> .	
[11]	AV	Address syndrome. Address syndrome injection. The value is: <b>0b0</b> When an injected error is recorded, the node either sets ERXADDR_EL1 and ERXSTATUS_EL1.AV for the access, or leaves these unchanged.	
[10]	PN	Poison flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.PN status flag. The value is: <b>0b0</b> When an injected error is recorded, the node sets ERXSTATUS_EL1.PN to 0.	

Bits	Name	Description	Reset
[9]	ER	Error Reported flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.ER status flag. The value is:  <b>0b0</b>  When an injected error is recorded, the node sets ERXSTATUS_EL1.ER according to the architecture-defined rules for setting the ER bit.	
[8]	CI	Critical Error flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.CI status flag. The value is:  <b>0b0</b>  The node does not support this type of flag  This behavior replaces the architecture-defined rules for setting the CI bit.	
[7:6]	CE	Corrected Error generation. The value is:  <b>0b01</b>  The fault generation feature of the node allows generation of a non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == 0b10.  All other values are reserved.	
[5]	DE	Deferred Error generation. The value is:  <b>0b1</b>  The fault generation feature of the node allows generation of this type of error.	
[4]	UEO	Latent or Restartable Error generation. The value is:  <b>0b0</b>  The fault generation feature of the node cannot generate this type of error.	
[3]	UER	Signaled or Recoverable Error generation. The value is:  <b>0b0</b>  The fault generation feature of the node cannot generate this type of error.	
[2]	UEU	Unrecoverable Error generation. The value is:  <b>0b0</b>  The fault generation feature of the node cannot generate this type of error.	
[1]	UC	Uncontainable Error generation. The value is:  <b>0b1</b>  The fault generation feature of the node allows generation of this type of error.	
[0]	OF	Overflow flag. Describes how the fault generation feature of the node sets the ERXSTATUS_EL1.OF status flag. The value is:  <b>0b0</b>  When an injected error is recorded, the node sets ERXSTATUS_EL1.OF according to the architecture-defined rules for setting the OF bit.	

## Access

MRS <Xt>, ERXPFGF\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXPFGF_EL1	0b11	0b000	0b0101	0b0100	0b100

## Accessibility

MRS <Xt>, ERXPFGF\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXPFGF_EL1 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGF_EL1;
elseif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGF_EL1;
elseif PSTATE.EL == EL3 then
    return ERXPFGF_EL1;

```

## A.10.8 ERXPFGCTL\_EL1, Selected Pseudo-fault Generation Control register

Accesses ext-ERR<n>PFGCTL for the error record <n> selected by AArch64-ERRSELR\_EL1.SEL.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

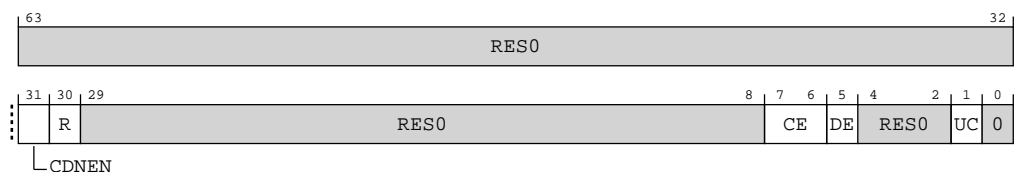
RAS

#### Reset value

0x0

### Bit descriptions

Figure A-119: AArch64\_erxpfctl\_el1 bit assignments



**Table A-323: ERXPFPGCTL\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31]	CDNEN	<p>Countdown Enable. Controls transfers from the value that is held in the ERXPFPGCDN_EL1 into the Error Generation Counter and enables this counter.</p> <p><b>0b0</b></p> <p>The Error Generation Counter is disabled.</p> <p><b>0b1</b></p> <p>The Error Generation Counter is enabled. On a write of 0b1 to this bit, the Error Generation Counter is set to ERXPFPGCDN_EL1.CDN.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0x0
[30]	R	<p>Restart. Controls whether, upon reaching zero, the Error Generation Counter restarts from the ERXPFPGCDN_EL1 value or stops.</p> <p><b>0b0</b></p> <p>On reaching 0, the Error Generation Counter will stop.</p> <p><b>0b1</b></p> <p>On reaching 0, the Error Generation Counter is set to ERXPFPGCDN_EL1.CDN.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0x0
[29:8]	RES0	Reserved	0x0
[7:6]	CE	<p>Corrected Error generation enable. Controls the type of Corrected Error condition that might be generated. The possible values are:</p> <p><b>0b00</b></p> <p>No error of this type will be generated.</p> <p><b>0b01</b></p> <p>A non-specific Corrected Error, that is, a Corrected Error that is recorded as ERXSTATUS_EL1.CE == 0b10, might be generated when the Error Generation Counter decrements to zero.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0x0
[5]	DE	<p>Deferred Error generation enable. The possible values are:</p> <p><b>0b0</b></p> <p>No error of this type will be generated.</p> <p><b>0b1</b></p> <p>An error of this type might be generated when the Error Generation Counter decrements to zero.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0x0
[4:2]	RES0	Reserved	0b000
[1]	UC	<p>Uncontainable Error generation enable. The possible values are:</p> <p><b>0b0</b></p> <p>No error of this type will be generated.</p> <p><b>0b1</b></p> <p>An error of this type might be generated when the Error Generation Counter decrements to zero.</p> <p>Cold reset only. Unaffected by Warm reset</p>	0b0
[0]	RES0	Reserved	0b0

## Access

MRS <Xt>, ERXPFPGCTL\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXPFPGCTL_EL1	0b11	0b000	0b0101	0b0100	0b101

MSR ERXPFPGCTL\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXPFPGCTL_EL1	0b11	0b000	0b0101	0b0100	0b101

## Accessibility

MRS <Xt>, ERXPFPGCTL\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEN == '1' && HFGTR_EL2.ERXPFPGCTL_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFPGCTL_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFPGCTL_EL1;
elsif PSTATE.EL == EL3 then
    return ERXPFPGCTL_EL1;

```

MSR ERXPFPGCTL\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEN == '1' && HFGWTR_EL2.ERXPFPGCTL_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFPGCTL_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFPGCTL_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXPFPGCTL_EL1 = X[t];

```



## A.10.9 ERXPFGCDN\_EL1, Selected Pseudo-fault Generation Countdown register

Accesses ext-ERR<n>PFGCDN for the error record <n> selected by AArch64-ERRSELR\_EL1.SEL.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

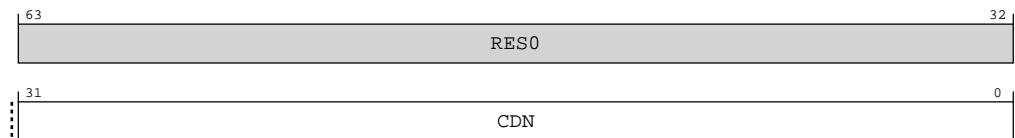
RAS

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure A-120: AArch64\_erxpfgcdn\_el1 bit assignments**



**Table A-326: ERXPFGCDN\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0x0
[31:0]	CDN	<p>Countdown value.</p> <p>This field is copied to Error Generation Counter when either:</p> <ul style="list-style-type: none"> <li>Software writes ERXPFGCTL_EL1.CDNEN with 1.</li> <li>The Error Generation Counter decrements to zero and ERXPFGCTL_EL1.R == 0b1.</li> </ul> <p>Unaffected by Cold or Warm reset.</p> <p><b>Note:</b> The current Error Generation Counter value is not visible to software.</p>	

### Access

MRS <Xt>, ERXPFGCDN\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXPFGCDN_EL1	0b11	0b000	0b0101	0b0100	0b110

MSR ERXPFGCDN\_EL1, &lt;Xt&gt;

<systemreg>	op0	op1	CRn	CRm	op2
ERXPFGCDN_EL1	0b11	0b000	0b0101	0b0100	0b110

## Accessibility

MRS &lt;Xt&gt;, ERXPFGCDN\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXPFGCDN_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGCDN_EL1;
elseif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXPFGCDN_EL1;
elseif PSTATE.EL == EL3 then
    return ERXPFGCDN_EL1;

```

MSR ERXPFGCDN\_EL1, &lt;Xt&gt;

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXPFGCDN_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCDN_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if SCR_EL3.FIEN == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXPFGCDN_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ERXPFGCDN_EL1 = X[t];

```

## A.10.10 ERXMISCO\_EL1, Selected Error Record Miscellaneous Register 0

Accesses ext-ERR<n>MISCO for the error record <n> selected by AArch64-ERRSELR\_EL1.SEL.

## Configurations

This register is available in all configurations.

**Attributes****Width**

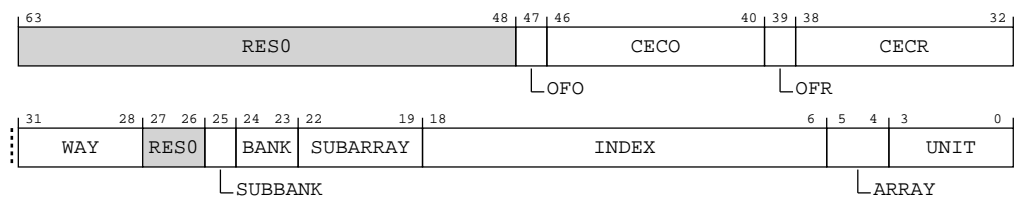
64

**Functional group**

RAS

**Reset value**

See individual bit resets.

**Bit descriptions****Figure A-121: AArch64\_erxmisc0\_el1 bit assignments****Table A-329: ERXMISCO\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	0x0
[47]	OFO	<p>Sticky overflow bit, other. Set to 1 when ERXMISCO_EL1.CECO is incremented and wraps through zero.</p> <p><b>0b0</b> Other counter has not overflowed.</p> <p><b>0b1</b> Other counter has overflowed.</p> <p>A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an UNKNOWN value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an UNKNOWN value.</p> <p>Unaffected by Cold or Warm reset.</p>	
[46:40]	CECO	<p>Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERXMISCO_EL1.CECR.</p> <p>Unaffected by Cold or Warm reset.</p>	
[39]	OFR	<p>Sticky overflow bit, repeat. Set to 1 when ERXMISCO_EL1.CECR is incremented and wraps through zero.</p> <p><b>0b0</b> Repeat counter has not overflowed.</p> <p><b>0b1</b> Repeat counter has overflowed.</p> <p>A direct write that modifies this bit might indirectly set ERXSTATUS_EL1.OF to an UNKNOWN value and a direct write to ERXSTATUS_EL1.OF that clears it to zero might indirectly set this bit to an UNKNOWN value.</p> <p>Unaffected by Cold or Warm reset.</p>	

Bits	Name	Description	Reset
[38:32]	CECR	<p>Corrected error count, repeat. Incremented for the first countable error, which also records other syndrome for the error, and subsequently for each countable error that matches the recorded other syndrome.</p> <p>This field resets to an <b>IMPLEMENTATION DEFINED</b> which might be UNKNOWN on a Cold reset. If the reset value is UNKNOWN, then the value of this field remains UNKNOWN until software initializes it.</p> <p>Unaffected by Cold or Warm reset.</p>	
[31:28]	WAY	<p>The encoding is dependent on the unit from which the error being recorded was detected. The possible values are:</p> <p>[L1 Data Cache]</p> <ul style="list-style-type: none"> <li>Indicates which Tag RAM way or data RAM way detected the error. Upper 2 bits are unused.</li> </ul> <p>[L2 TLB]</p> <ul style="list-style-type: none"> <li>Indicates which RAM detected an error. The possible values are 0 (RAM 1) to 9 (RAM 10).</li> </ul> <p>[L1 Instruction Cache]</p> <ul style="list-style-type: none"> <li>Indicates which way detected the error. Upper 2 bits are unused.</li> </ul> <p>[L2 Cache]</p> <ul style="list-style-type: none"> <li>Indicates which way detected the error. Upper 1 bit unused.</li> </ul> <p>Unaffected by Cold or Warm reset.</p>	
[27:26]	RES0	Reserved	0b00
[25]	SUBBANK	<p>The encoding is dependent on the unit from which the error being recorded was detected. The possible values are:</p> <p>[L1 Instruction Cache]</p> <ul style="list-style-type: none"> <li>Indicates which subbank detected the error, valid for Instruction Data Cache. For Tag errors this field is zero.</li> </ul> <p>Unaffected by Cold or Warm reset.</p>	
[24:23]	BANK	<p>The encoding is dependent on the unit from which the error being recorded was detected. The possible values are:</p> <p>[L2 Cache]</p> <ul style="list-style-type: none"> <li>Indicates which L2 bank detected the error. Upper 1 bit is unused.</li> </ul> <p>[L1 Instruction Cache]</p> <ul style="list-style-type: none"> <li>Indicates which bank detected the error, valid for Instruction Data Cache. For Tag errors this field is zero.</li> </ul> <p>Unaffected by Cold or Warm reset.</p>	

Bits	Name	Description	Reset
[22:19]	SUBARRAY	<p>The encoding is dependent on the unit from which the error being recorded was detected. The possible values are:</p> <p>[L2 Cache]</p> <ul style="list-style-type: none"> <li>Indicates which L2 data doubleword detected the error. Upper 1 bit is unused.</li> </ul> <p>[L1 Data Cache]</p> <ul style="list-style-type: none"> <li>Indicates for L1 Data RAM which word had the error detected. For L1 Tag RAMs which bank had the error (0b0000: bank0 , 0b0001: bank1)</li> </ul> <p>Unaffected by Cold or Warm reset.</p>	
[18:6]	INDEX	<p>The encoding is dependent on the unit from which the error being recorded was detected. The possible values are:</p> <p>[L2 Cache]</p> <ul style="list-style-type: none"> <li>Indicates which index detected the error. Upper bits of the index are unused depending on the cache size.</li> </ul> <p>[L1 Data Cache]</p> <ul style="list-style-type: none"> <li>Indicates which index detected the error. Upper bits of the index are unused depending on the cache size</li> </ul> <p>[L2 TLB]</p> <ul style="list-style-type: none"> <li>Index of TLB RAM. Upper 4 bits are unused.</li> </ul> <p>[L1 Instruction Cache]</p> <ul style="list-style-type: none"> <li>Indicates which index detected the error. Upper bits of the index are unused depending on the cache size.</li> </ul> <p>Unaffected by Cold or Warm reset.</p>	

Bits	Name	Description	Reset
[5:4]	ARRAY	<p>The encoding is dependent on the unit from which the error being recorded was detected. The possible values are:</p> <p>[L2 Cache]</p> <p>Indicates which array detected the error. The possible values are:</p> <ul style="list-style-type: none"> <li>0b00 L2 Tag RAM.</li> <li>0b01 L2 Data RAM.</li> <li>0b10 L2 TQ data RAM.</li> <li>0b11 CHI Error.</li> </ul> <p>[L1 Data Cache]</p> <p>Indicates which array detected the error. The possible values are:</p> <ul style="list-style-type: none"> <li>00 LS Tag RAM 0.</li> <li>01 LS Tag RAM 1.</li> <li>10 LS Data RAM.</li> <li>11 LS Tag RAM 2.</li> </ul> <p>[L1 Instruction Cache]</p> <p>Indicates which array that detected the error, Data Array has higher priority. The possible values are:</p> <ul style="list-style-type: none"> <li>0b00 Tag.</li> <li>0b01 Data.</li> <li>0b10 Macro-OP cache.</li> </ul> <p>Unaffected by Cold or Warm reset.</p>	
[3:0]	UNIT	<p>Indicates the unit which detected the error. The possible values are:</p> <p><b>0b0001</b> L1 Instruction Cache.</p> <p><b>0b0010</b> L2 TLB.</p> <p><b>0b0100</b> L1 Data Cache.</p> <p><b>0b1000</b> L2 Cache.</p>	

## Access

MRS <Xt>, ERXMISCO\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISCO_EL1	0b11	0b000	0b0101	0b0101	0b000

MSR ERXMISCO\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISCO_EL1	0b11	0b000	0b0101	0b0101	0b000

## Accessibility

MRS <Xt>, ERXMISCO\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXMISCn_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISCO_EL1;
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISCO_EL1;
elseif PSTATE.EL == EL3 then
    return ERXMISCO_EL1;

```

MSR ERXMISCO\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISCO_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISCO_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ERXMISCO_EL1 = X[t];

```

## A.10.11 ERXMISC1\_EL1, Selected Error Record Miscellaneous Register 1

Accesses ext-ERR<n>MISC1 for the error record <n> selected by AArch64-ERRSELR\_EL1.SEL.

### Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

0x0

Bit descriptions

Figure A-122: AArch64\_erxmisc1\_el1 bit assignments

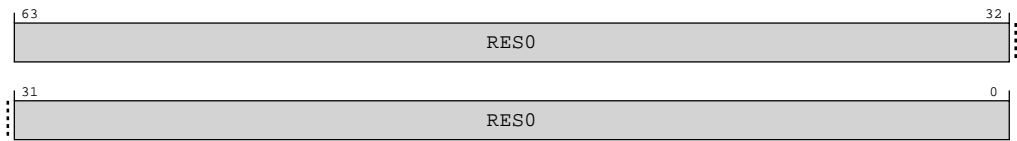


Table A-332: ERXMISC1\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

Access

MRS <Xt>, ERXMISC1\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC1_EL1	0b11	0b000	0b0101	0b0101	0b001

MSR ERXMISC1\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC1_EL1	0b11	0b000	0b0101	0b0101	0b001

Accessibility

MRS <Xt>, ERXMISC1\_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXMISCn_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC1_EL1;
elsif PSTATE.EL == EL2 then
```



```

    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC1_EL1;
elseif PSTATE.EL == EL3 then
    return ERXMISC1_EL1;

```

MSR ERXMISC1\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC1_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC1_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ERXMISC1_EL1 = X[t];

```

## A.10.12 ERXMISC2\_EL1, Selected Error Record Miscellaneous Register 2

Accesses ext-ERR<n>MISC2 for the error record <n> selected by AArch64-ERRSELR\_EL1.SEL.

### Configurations

This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

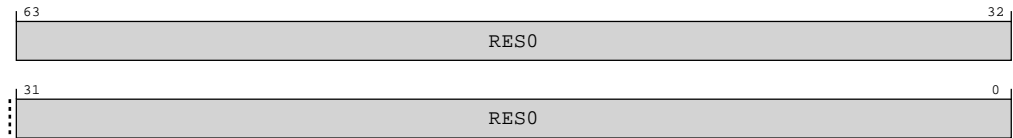
RAS

#### Reset value

0x0

## Bit descriptions

**Figure A-123: AArch64\_erxmisc2\_el1 bit assignments**



**Table A-335: ERXMISC2\_EL1 bit descriptions**

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

## Access

MRS <Xt>, ERXMISC2\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC2_EL1	0b11	0b000	0b0101	0b0101	0b010

MSR ERXMISC2\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC2_EL1	0b11	0b000	0b0101	0b0101	0b010

## Accessibility

MRS <Xt>, ERXMISC2\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXMISCN_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC2_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC2_EL1;
elsif PSTATE.EL == EL3 then
    return ERXMISC2_EL1;

```

MSR ERXMISC2\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then

```

```
if EL2Enabled() && HCR_EL2.TERR == '1' then
    AArch64.SystemAccessTrap(EL2, 0x18);
elseif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
then
    AArch64.SystemAccessTrap(EL2, 0x18);
elseif SCR_EL3.TERR == '1' then
    AArch64.SystemAccessTrap(EL3, 0x18);
else
    ERXMISC2_EL1 = X[t];
elseif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC2_EL1 = X[t];
elseif PSTATE.EL == EL3 then
    ERXMISC2_EL1 = X[t];
```

A.10.13 ERXMISC3\_EL1, Selected Error Record Miscellaneous Register 3

Accesses ext-ERR<n>MISC3 for the error record <n> selected by AArch64-ERRSELR\_EL1.SEL.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

RAS

Reset value

0x0

Bit descriptions

Figure A-124: AArch64\_erxmisc3\_el1 bit assignments

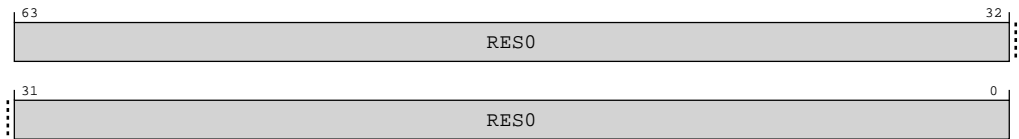


Table A-338: ERXMISC3\_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0x0

Access

MRS <Xt>, ERXMISC3\_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC3_EL1	0b11	0b000	0b0101	0b0101	0b011

MSR ERXMISC3\_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
ERXMISC3_EL1	0b11	0b000	0b0101	0b0101	0b011

## Accessibility

MRS <Xt>, ERXMISC3\_EL1

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGTR_EL2.ERXMISCn_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC3_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return ERXMISC3_EL1;
elsif PSTATE.EL == EL3 then
    return ERXMISC3_EL1;

```

MSR ERXMISC3\_EL1, <Xt>

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TERR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && SCR_EL3.FGTEn == '1' && HFGWTR_EL2.ERXMISCn_EL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC3_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TERR == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        ERXMISC3_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    ERXMISC3_EL1 = X[t];

```

# Appendix B External registers

This appendix contains the descriptions for the Cortex®-X2 external registers.

## B.1 CoreROM register summary

The summary table provides an overview of all CoreROM registers in the core. Individual register descriptions provide detailed information.

**Table B-1: CoreROM register summary**

Offset	Name	Reset	Width	Description
0x000	COREROM_ROMENTRY0	See individual bit resets.	32-bit	Core ROM table Entry 0
0x004	COREROM_ROMENTRY1	See individual bit resets.	32-bit	Core ROM table Entry 1
0x008	COREROM_ROMENTRY2	See individual bit resets.	32-bit	Core ROM table Entry 2
0x00C	COREROM_ROMENTRY3	See individual bit resets.	32-bit	Core ROM table Entry 3
0xFB8	COREROM_AUTHSTATUS	See individual bit resets.	32-bit	Core ROM table Authentication Status Register
0xFBC	COREROM_DEVARCH	See individual bit resets.	32-bit	Core ROM table Device Architecture Register
0xFCC	COREROM_DEVTYPE	See individual bit resets.	32-bit	Core ROM table Device Type Register
0xFD0	COREROM_PIDR4	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 4
0xFE0	COREROM_PIDR0	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 0
0xFE4	COREROM_PIDR1	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 1
0xFE8	COREROM_PIDR2	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 2
0xFEC	COREROM_PIDR3	See individual bit resets.	32-bit	Core ROM table Peripheral Identification Register 3
0xFF0	COREROM_CIDR0	See individual bit resets.	32-bit	Core ROM table Component Identification Register 0
0xFF4	COREROM_CIDR1	See individual bit resets.	32-bit	Core ROM table Component Identification Register 1
0xFF8	COREROM_CIDR2	See individual bit resets.	32-bit	Core ROM table Component Identification Register 2
0xFFC	COREROM_CIDR3	See individual bit resets.	32-bit	Core ROM table Component Identification Register 3

### B.1.1 COREROM\_ROMENTRY0, Core ROM table Entry 0

Provides the address offset for one CoreSight component.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

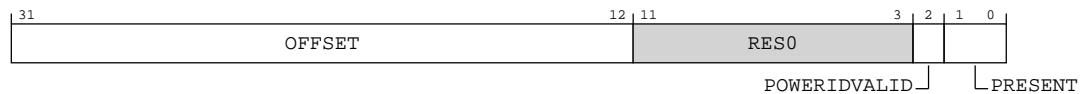
CoreROM

**Register offset**

0x000

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-1: ext\_COREROM\_ROMENTRY0 bit assignments****Table B-2: COREROM\_ROMENTRY0 bit descriptions**

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation:  Component Address = ROM Table Base Address + (OFFSET << 12).  <b>0b00000000000000000000000000000000</b> Core DBG component at address 0x1_0000.	
[11:3]	RES0	Reserved	0b0
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID.  <b>0b0</b> A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table.  <b>0b11</b> The ROM Entry is present.	

**B.1.2 COREROM\_ROMENTRY1, Core ROM table Entry 1**

Provides the address offset for one CoreSight component.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

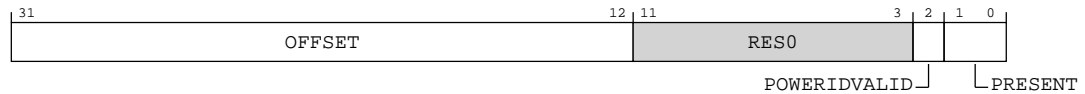
CoreROM

**Register offset**

0x004

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-2: ext\_COREROM\_ROMENTRY1 bit assignments****Table B-3: COREROM\_ROMENTRY1 bit descriptions**

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation:  Component Address = ROM Table Base Address + (OFFSET << 12).  <b>0b00000000000000000000000000000000</b> CORE PMU component at address 0x2_0000.	
[11:3]	RES0	Reserved	0b0
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID.  <b>0b0</b> A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table.  <b>0b11</b> The ROM Entry is present.	

**B.1.3 COREROM\_ROMENTRY2, Core ROM table Entry 2**

Provides the address offset for one CoreSight component.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

CoreROM

**Register offset**

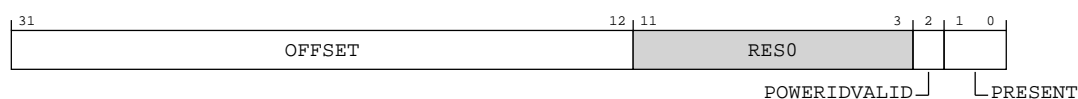
0x008

**Reset value**

See individual bit resets.

## Bit descriptions

**Figure B-3: ext\_COREROM\_ROMENTRY2 bit assignments**



### Table B-4: COREROM\_ROMENTRY2 bit descriptions

Bits	Name	Description	Reset
[31:12]	OFFSET	<p>The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation:</p> $\text{Component Address} = \text{ROM Table Base Address} + (\text{OFFSET} \ll 12).$ <p><b>0b000000000000000110000</b></p> <p>Core trace unit component at address 0x3_0000.</p>	
[11:3]	RES0	Reserved	0b0
[2]	POWERIDVALID	<p>Indicates if the Power domain ID field contains a Power domain ID.</p> <p><b>0b0</b></p> <p>A power domain ID is not provided.</p>	
[1:0]	PRESENT	<p>Indicates whether an entry is present at this location in the ROM Table.</p> <p><b>0b11</b></p> <p>The ROM Entry is present.</p>	

### B.1.4 COREROM\_ROMENTRY3, Core ROM table Entry 3

Provides the address offset for one CoreSight component.

## Configurations

- This register is available in all configurations.

## Attributes

## Width

32

### Functional group

CoreROM

## Register offset

0x00C

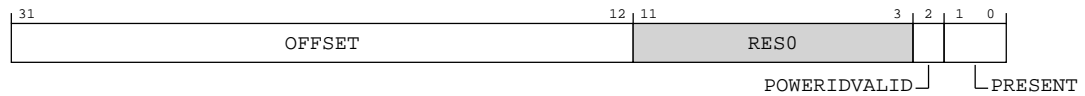
## Reset value

See individual bit resets.



## Bit descriptions

**Figure B-4: ext\_COREROM\_ROMENTRY3 bit assignments**



**Table B-5: COREROM\_ROMENTRY3 bit descriptions**

Bits	Name	Description	Reset
[31:12]	OFFSET	The component address, relative to the base address of this ROM Table. The component address is calculated using the following equation:  Component Address = ROM Table Base Address + (OFFSET << 12).  <b>0b000000000000001000000</b>  Core ELA component at address 0x4_0000. When the core is configured without ELA, this field is set to 0x000.	
[11:3]	RES0	Reserved	0b0
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID.  <b>0b0</b>  A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table.  <b>0b11</b>  The ROM Entry is present. When the core is configured without ELA, this field is set to 0x0.	

## B.1.5 COREROM\_AUTHSTATUS, Core ROM table Authentication Status Register

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

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

CoreROM

#### Register offset

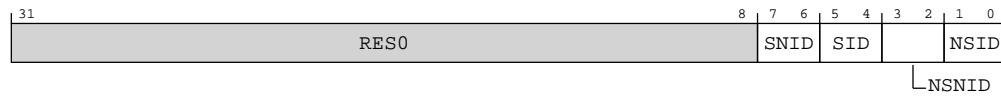
0xFB8

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-5: ext\_COREROM\_AUTHSTATUS bit assignments**



**Table B-6: COREROM\_AUTHSTATUS bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:6]	SNID	Secure Non-invasive Debug.	
[5:4]	SID	Secure Invasive Debug.  <b>0b10</b> Secure invasive debug disabled. ExternalSecureInvasiveDebugEnabled() == FALSE.  <b>0b11</b> Secure invasive debug enabled. ExternalSecureInvasiveDebugEnabled() == TRUE.	
[3:2]	NSNID	Non-secure Non-invasive Debug.  <b>0b00</b> Debug level is not supported.	
[1:0]	NSID	Non-secure Invasive Debug.  <b>0b00</b> Debug level is not supported.	

## B.1.6 COREROM\_DEVARCH, Core ROM table Device Architecture Register

Identifies the architect and architecture of a CoreSight component.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

CoreROM

#### Register offset

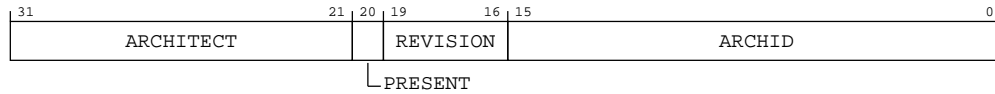
0xFBC

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-6: ext\_COREROM\_DEVARCH bit assignments**



**Table B-7: COREROM\_DEVARCH bit descriptions**

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. <b>0b01000111011</b> JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.	
[20]	PRESENT	Present. <b>0b1</b> DEVARCH information present.	
[19:16]	REVISION	Revision. <b>0b0000</b> Revision 0.	
[15:0]	ARCHID	Architecture ID. <b>0b0000101011110111</b> ROM Table v0. The debug tool must inspect ext-COREROM_DEVTYPE and ext-COREROM_DEVID to determine further information about the ROM Table.	

### B.1.7 COREROM\_DEVTYPE, Core ROM table Device Type Register

A debugger can use DEVTYPE to obtain information about a component that has an unrecognized part number.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

CoreROM

##### Register offset

0xFCC

##### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-7: ext\_COREROM\_DEVTYPE bit assignments**



**Table B-8: COREROM\_DEVTYPE bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	SUB	Sub number <b>0b0000</b> Other, undefined.	
[3:0]	MAJOR	Major number <b>0b0000</b> Miscellaneous.	

## B.1.8 COREROM\_PIDR4, Core ROM table Peripheral Identification Register 4

Provides CoreSight discovery information.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

CoreROM

#### Register offset

0xFD0

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-8: ext\_COREROM\_PIDR4 bit assignments**



**Table B-9: COREROM\_PIDR4 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	SIZE	4KB count.  <b>0b0000</b> The component uses a single 4KB block.	
[3:0]	DES_2	JEP106 continuation code.  <b>0b0100</b> Arm Limited. Number of 0x7F bytes in full JEP106 code 0x7F 0x7F 0x7F 0x7F 0x3B.	

## B.1.9 COREROM\_PIDR0, Core ROM table Peripheral Identification Register 0

Provides CoreSight discovery information.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

CoreROM

#### Register offset

0xFE0

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-9: ext\_COREROM\_PIDR0 bit assignments****Table B-10: COREROM\_PIDR0 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PART_0	Part number bits [7:0].  <b>0b01000111</b> Cortex®-X2 Core ROM table. Bits [7:0] of part number 0xD47.	

## B.1.10 COREROM\_PIDR1, Core ROM table Peripheral Identification Register 1

Provides CoreSight discovery information.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

CoreROM

#### Register offset

0xFE4

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-10: ext\_COREROM\_PIDR1 bit assignments**



**Table B-11: COREROM\_PIDR1 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	DES_0	JEP106 identification code bits [3:0]. <b>0b1011</b> Arm Limited. Bits [3:0] of JEP106 identification code 0x3B.	
[3:0]	PART_1	Part number bits [11:8]. <b>0b1101</b> Cortex®-X2 Core ROM table. Bits [11:8] of part number 0xD47.	

## B.1.11 COREROM\_PIDR2, Core ROM table Peripheral Identification Register 2

Provides CoreSight discovery information.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

CoreROM

#### Register offset

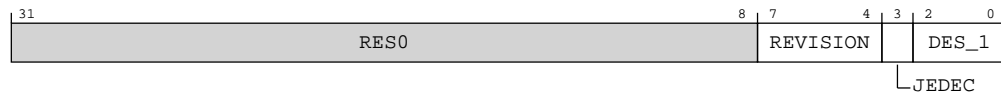
0xFE8

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-11: ext\_COREROM\_PIDR2 bit assignments**



**Table B-12: COREROM\_PIDR2 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. <b>0b0010</b> Revision r2p1.	
[3]	JEDEC	JEDEC assignee. <b>0b1</b> JEDEC-assignee values is used.	
[2:0]	DES_1	JEP106 identification code bits [6:4]. <b>0b011</b> Arm Limited. Bits [6:4] of JEP106 identification code 0x3B.	

### B.1.12 COREROM\_PIDR3, Core ROM table Peripheral Identification Register 3

Provides CoreSight discovery information.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

CoreROM

##### Register offset

0xFEC

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-12: ext\_COREROM\_PIDR3 bit assignments**



**Table B-13: COREROM\_PIDR3 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	REVAND	Part minor revision. Parts using ext-EDPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. <b>0b0000</b>	
[3:0]	CMOD	Customer Modified. <b>0b0000</b> The component is not modified from the original design.	

### B.1.13 COREROM\_CIDR0, Core ROM table Component Identification Register 0

Provides CoreSight discovery information.

#### Configurations

- This register is available in all configurations.



## Attributes

## Width

32

### Functional group

CoreROM

## Register offset

0xFF0

## Reset value

See individual bit resets.

## Bit descriptions

### Figure B-13: ext\_COREROM\_CIDR0 bit assignments



### Table B-14: COREROM\_CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_0	CoreSight component identification preamble. <b>0b00001101</b> CoreSight component identification preamble.	

### B.1.14 COREROM\_CIDR1, Core ROM table Component Identification Register 1

Provides CoreSight discovery information.

## Configurations

- This register is available in all configurations.

## Attributes

## Width

32

### Functional group

CoreROM

## Register offset

0xFF4

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-14: ext\_COREROM\_CIDR1 bit assignments****Table B-15: COREROM\_CIDR1 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	CLASS	CoreSight component class. <b>0b1001</b> CoreSight component.	
[3:0]	PRMBL_1	CoreSight component identification preamble. <b>0b0000</b> CoreSight component identification preamble.	

## B.1.15 COREROM\_CIDR2, Core ROM table Component Identification Register 2

Provides CoreSight discovery information.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

CoreROM

**Register offset**

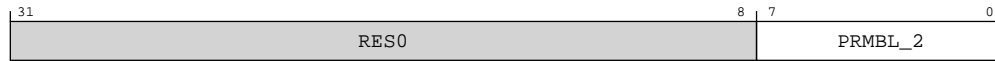
0xFF8

**Reset value**

See individual bit resets.

## Bit descriptions

**Figure B-15: ext\_COREROM\_CIDR2 bit assignments**



**Table B-16: COREROM\_CIDR2 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_2	CoreSight component identification preamble. <b>0b00000101</b> CoreSight component identification preamble.	

## B.1.16 COREROM\_CIDR3, Core ROM table Component Identification Register 3

Provides CoreSight discovery information.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

CoreROM

#### Register offset

0xFFC

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-16: ext\_COREROM\_CIDR3 bit assignments**



**Table B-17: COREROM\_CIDR3 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[7:0]	PRMBL_3	CoreSight component identification preamble.  <b>0b10110001</b> CoreSight component identification preamble.	

## B.2 PPM register summary

The summary table provides an overview of all PPM registers in the core. Individual register descriptions provide detailed information.

**Table B-18: PPM register summary**

Offset	Name	Reset	Width	Description
0x000	CPUPPMCR	See individual bit resets.	64-bit	Power Performance Management Register
0x010	CPUPPMCR2	See individual bit resets.	64-bit	Power Performance Management Register
0x020	CPUPPMCR3	See individual bit resets.	64-bit	Power Performance Management Register
0x080	CPUPPMCR4	See individual bit resets.	64-bit	Power Performance Management Register
0x088	CPUPPMCR5	See individual bit resets.	64-bit	Power Performance Management Register
0x090	CPUPPMCR6	See individual bit resets.	64-bit	Power Performance Management Register

### B.2.1 CPUPPMCR, Power Performance Management Register

This register contains control bits that affect the CPU behavior

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

PPM

##### Register offset

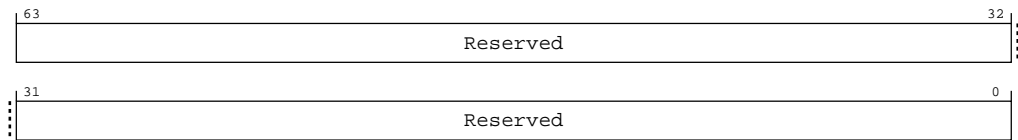
0x000

##### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-17: ext\_CPUPPMCR bit assignments**



**Table B-19: CPUPPMCR bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	None	

## B.2.2 CPUPPMCR2, Power Performance Management Register

This register contains control bits that affect the CPU behavior

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

PPM

#### Register offset

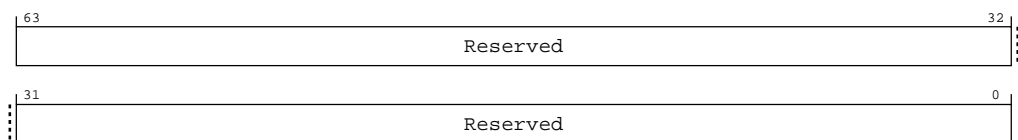
0x010

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-18: ext\_CPUPPMCR2 bit assignments**



**Table B-20: CPUPPMCR2 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	None	

### B.2.3 CPUPPMCR3, Power Performance Management Register

This register contains control bits that affect the CPU behavior

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

PPM

##### Register offset

0x020

##### Reset value

See individual bit resets.

#### Bit descriptions

Figure B-19: ext\_CPUPPMCR3 bit assignments

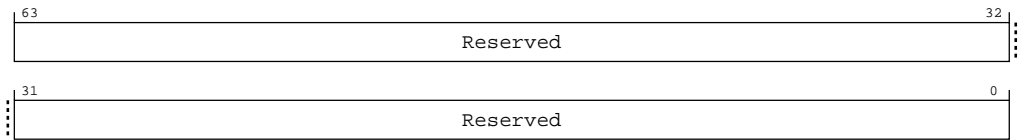


Table B-21: CPUPPMCR3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	

### B.2.4 CPUPPMCR4, Power Performance Management Register

This register contains control bits that affect the CPU behavior

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

64

Functional group

PPM

Register offset

0x080

Reset value

See individual bit resets.

Bit descriptions

Figure B-20: ext\_CPUPPMCR4 bit assignments

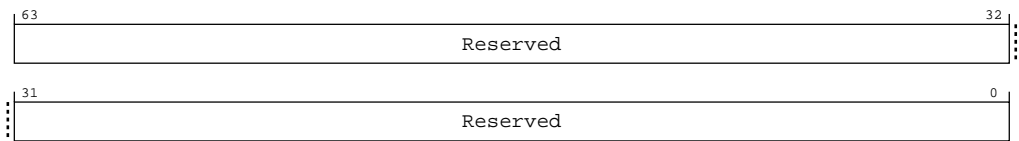


Table B-22: CPUPPMCR4 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	None	

B.2.5 CPUPPMCR5, Power Performance Management Register

This register contains control bits that affect the CPU behavior

Configurations

- This register is available in all configurations.

Attributes

Width

64

Functional group

PPM

Register offset

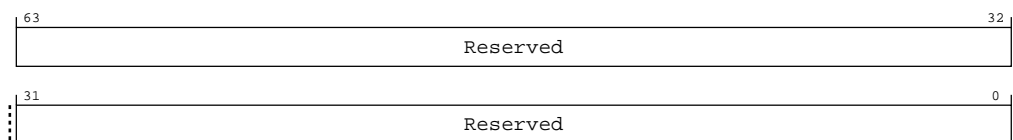
0x088

Reset value

See individual bit resets.

## Bit descriptions

**Figure B-21: ext\_CPUPPMCR5 bit assignments**



**Table B-23: CPUPPMCR5 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	None	

## B.2.6 CPUPPMCR6, Power Performance Management Register

This register contains control bits that affect the CPU behavior

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

PPM

#### Register offset

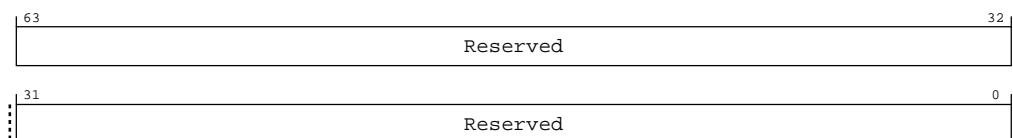
0x090

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-22: ext\_CPUPPMCR6 bit assignments**



**Table B-24: CPUPPMCR6 bit descriptions**

Bits	Name	Description	Reset
[63:0]	Reserved	None	



## B.3 PMU register summary

The summary table provides an overview of all PMU registers in the core. Individual register descriptions provide detailed information.

**Table B-25: PMU register summary**

Offset	Name	Reset	Width	Description
0x600	PMPCSSR	See individual bit resets.	64-bit	Snapshot Program Counter Sample Register
0x608	PMCIDSSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL1 Sample Register
0x60C	PMCID2SSR	See individual bit resets.	32-bit	Snapshot CONTEXTIDR_EL2 Sample Register
0x610	PMSSSR	0x1	32-bit	PMU Snapshot Status Register
0x618	PMCCNTSR	See individual bit resets.	64-bit	PMU Cycle Counter Snapshot Register
0x620	PMEVCNTSR0	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x628	PMEVCNTSR1	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x630	PMEVCNTSR2	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x638	PMEVCNTSR3	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x640	PMEVCNTSR4	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x648	PMEVCNTSR5	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x650	PMEVCNTSR6	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x658	PMEVCNTSR7	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x660	PMEVCNTSR8	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x668	PMEVCNTSR9	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x670	PMEVCNTSR10	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x678	PMEVCNTSR11	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x680	PMEVCNTSR12	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x688	PMEVCNTSR13	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x690	PMEVCNTSR14	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x698	PMEVCNTSR15	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6A0	PMEVCNTSR16	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6A8	PMEVCNTSR17	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6B0	PMEVCNTSR18	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6B8	PMEVCNTSR19	See individual bit resets.	64-bit	PMU Event Counter Snapshot Register
0x6F0	PMSSCR	See individual bit resets.	32-bit	PMU Snapshot Capture Register
0xE00	PMCFGR	See individual bit resets.	32-bit	Performance Monitors Configuration Register
0xE04	PMCR_ELO	See individual bit resets.	32-bit	Performance Monitors Control Register
0xE20	PMCEID0	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 0
0xE24	PMCEID1	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 1
0xE28	PMCEID2	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 2
0xE2C	PMCEID3	See individual bit resets.	32-bit	Performance Monitors Common Event Identification register 3
0xE40	PMMIR	See individual bit resets.	32-bit	Performance Monitors Machine Identification Register

Offset	Name	Reset	Width	Description
0xFBC	<a href="#">PMDEVARCH</a>	See individual bit resets.	32-bit	Performance Monitors Device Architecture register
0xFC8	<a href="#">PMDEVID</a>	See individual bit resets.	32-bit	Performance Monitors Device ID register
0xFCC	<a href="#">PMDEVTYPE</a>	See individual bit resets.	32-bit	Performance Monitors Device Type register
0xFD0	<a href="#">PMPIDR4</a>	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	<a href="#">PMPIDR0</a>	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	<a href="#">PMPIDR1</a>	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	<a href="#">PMPIDR2</a>	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	<a href="#">PMPIDR3</a>	See individual bit resets.	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	<a href="#">PMCIDR0</a>	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 0
0xFF4	<a href="#">PMCIDR1</a>	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 1
0xFF8	<a href="#">PMCIDR2</a>	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 2
0xFFC	<a href="#">PMCIDR3</a>	See individual bit resets.	32-bit	Performance Monitors Component Identification Register 3

### B.3.1 PMPCSSR, Snapshot Program Counter Sample Register

Captured copy of the Program Counter.

## Configurations

- This register is available in all configurations.

## Attributes

## Width

64

### Functional group

PMU

## Register offset

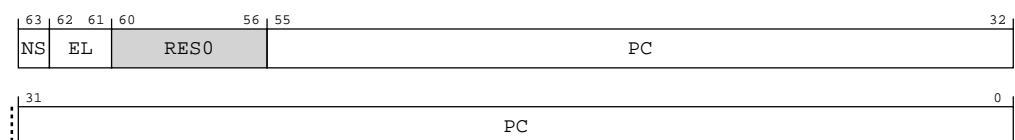
0x600

## Reset value

See individual bit resets.

## Bit descriptions

**Figure B-23: ext\_PMP\_CSSR bit assignments**



**Table B-26: PMPCSSR bit descriptions**

Bits	Name	Description	Reset
[63]	NS	Non-secure sample.  <b>0b0</b> The captured instruction was executed in Secure state.  <b>0b1</b> The captured instruction was executed in Non-secure state.	
[62:61]	EL	Exception level sample. The Exception level the captured instruction was executed at.	
[60:56]	RES0	Reserved	0b0
[55:0]	PC	Sampled PC.  The instruction address for the sampled instruction. The sampled instruction must be an instruction recently executed by the PE.  The architecture does not require that all instructions are eligible for sampling. However, it must be possible to reference instructions at branch targets. The branch target for a conditional branch instruction that fails its Condition code check is the instruction following the conditional branch target.  The sampled instruction must be architecturally executed. However, in exceptional circumstances, such as a change in security state or other boundary condition, it is permissible to sample an instruction that was speculatively executed and not architecturally executed.	

### B.3.2 PMCIDSSR, Snapshot CONTEXTIDR\_EL1 Sample Register

Captured copy of the CONTEXTIDR\_EL1 register.

The value captured must relate to the instruction captured in PMPCSSR.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

PMU

##### Register offset

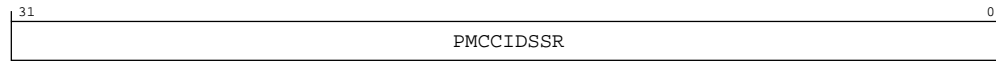
0x608

##### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-24: ext\_PMCIDSSR bit assignments**



**Table B-27: PMCIDSSR bit descriptions**

Bits	Name	Description	Reset
[31:0]	PMCCIDSSR	PMCIDSR sample. Sampled CONTEXTIDR_EL1 snapshot.	

## B.3.3 PMCID2SSR, Snapshot CONTEXTIDR\_EL2 Sample Register

Captured copy of the CONTEXTIDR\_EL2 register.

The value captured must relate to the instruction captured in PMPCSSR.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

PMU

#### Register offset

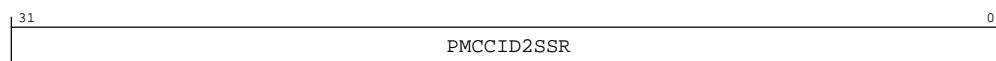
0x60C

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-25: ext\_PMCID2SSR bit assignments**



**Table B-28: PMCID2SSR bit descriptions**

Bits	Name	Description	Reset
[31:0]	PMCCID2SSR	PMCID2SR sample. Sampled CONTEXTIDR_EL2 snapshot.	

### B.3.4 PMSSSR, PMU Snapshot Status Register

Holds status information about the captured counters.

## Configurations

- This register is available in all configurations.

## Attributes

## Width

32

### Functional group

PMU

## Register offset

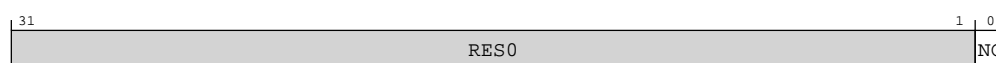
0x610

## Reset value

0x1

## Bit descriptions

**Figure B-26: ext\_PMSSSR bit assignments**



### Table B-29: PMSSSR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	0b0
[0]	NC	<p>No capture. Indicates whether the PMU counters have been captured.</p> <p><b>0b0</b> PMU counters captured.</p> <p><b>0b1</b> PMU counters not captured.</p>	0b1

### B.3.5 PMCCNTSR, PMU Cycle Counter Snapshot Register

Captured copy of PMCCNTR\_ELO. Once captured, the value in PMCCNTSR is unaffected by writes to PMCCNTR\_ELO and PMCR\_ELO.C.

## Configurations

- This register is available in all configurations.

Attributes

Width

64

Functional group

PMU

Register offset

0x618

Reset value

See individual bit resets.

Bit descriptions

Figure B-27: ext\_PMCCNTSR bit assignments

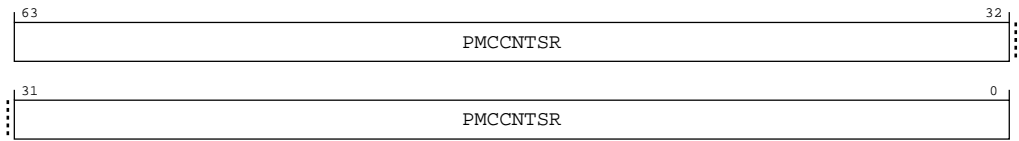


Table B-30: PMCCNTSR bit descriptions

Bits	Name	Description	Reset
[63:0]	PMCCNTSR	PMCCNTR_ELO sample. Sampled cycle count.	

B.3.6 PMEVCNTR0, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

Configurations

- This register is available in all configurations.

Attributes

Width

64

Functional group

PMU

Register offset

0x620

Reset value

See individual bit resets.

Bit descriptions

Figure B-28: ext\_PMEVCNTR19 bit assignments

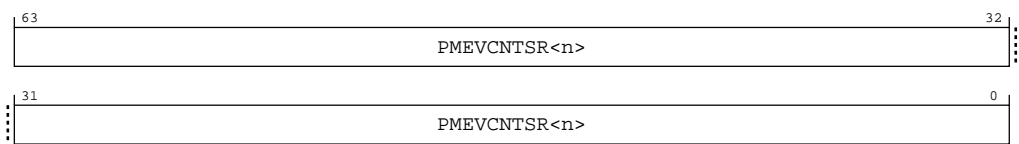


Table B-31: PMEVCNTR0 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

B.3.7 PMEVCNTR1, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

Configurations

- This register is available in all configurations.

Attributes

Width

64

Functional group

PMU

Register offset

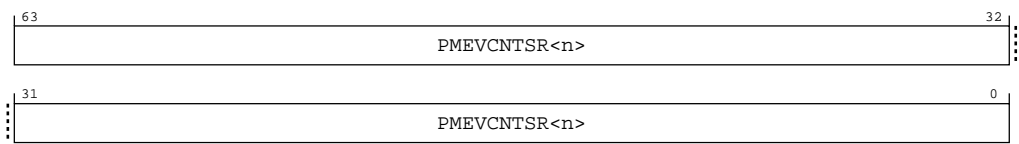
0x628

Reset value

See individual bit resets.

Bit descriptions

Figure B-29: ext\_PMEVCNTR19 bit assignments



**Table B-32: PMEVCNTR1 bit descriptions**

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

### B.3.8 PMEVCNTR2, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

64

##### Functional group

PMU

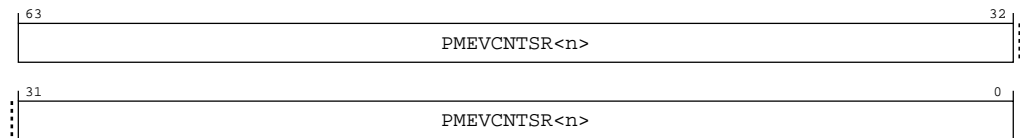
##### Register offset

0x630

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-30: ext\_PMEVCNTR19 bit assignments****Table B-33: PMEVCNTR2 bit descriptions**

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	



B.3.9 PMEVCNTR3, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

Configurations

- This register is available in all configurations.

Attributes

Width

64

Functional group

PMU

Register offset

0x638

Reset value

See individual bit resets.

Bit descriptions

Figure B-31: ext\_PMEVCNTR19 bit assignments

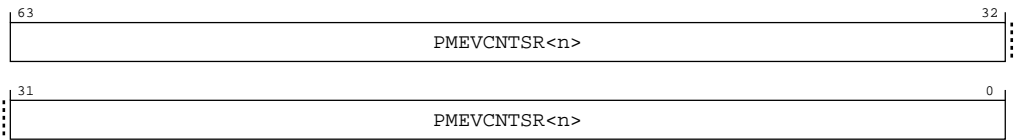


Table B-34: PMEVCNTR3 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

B.3.10 PMEVCNTR4, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

Configurations

- This register is available in all configurations.

Attributes

Width

64

Functional group

PMU

Register offset

0x640

Reset value

See individual bit resets.

Bit descriptions

Figure B-32: ext\_PMEVCNTR19 bit assignments

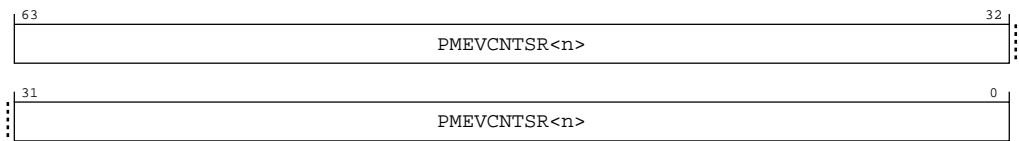


Table B-35: PMEVCNTR4 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

B.3.11 PMEVCNTR5, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

Configurations

- This register is available in all configurations.

Attributes

Width

64

Functional group

PMU

Register offset

0x648

Reset value

See individual bit resets.

Bit descriptions

Figure B-33: ext\_PMEVCNTR19 bit assignments

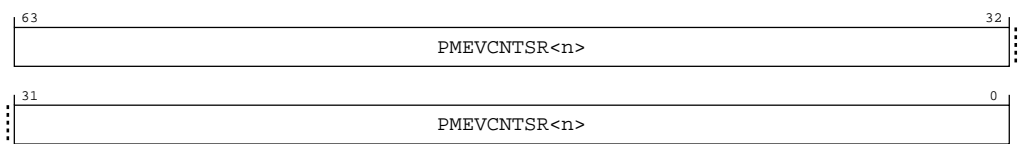


Table B-36: PMEVCNTR5 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

B.3.12 PMEVCNTR6, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

Attributes

Width

64

Functional group

PMU

Register offset

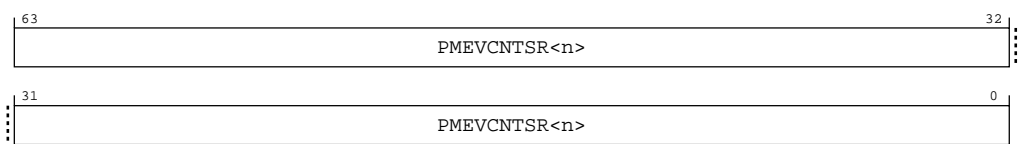
0x650

Reset value

See individual bit resets.

Bit descriptions

Figure B-34: ext\_PMEVCNTR19 bit assignments



**Table B-37: PMEVCNTR6 bit descriptions**

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

### B.3.13 PMEVCNTR7, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

#### Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

#### Attributes

##### Width

64

##### Functional group

PMU

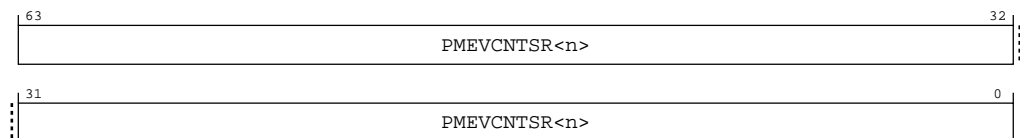
##### Register offset

0x658

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-35: ext\_PMEVCNTR19 bit assignments****Table B-38: PMEVCNTR7 bit descriptions**

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

B.3.14 PMEVCNTR8, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

Attributes

Width

64

Functional group

PMU

Register offset

0x660

Reset value

See individual bit resets.

Bit descriptions

Figure B-36: ext\_PMEVCNTR19 bit assignments

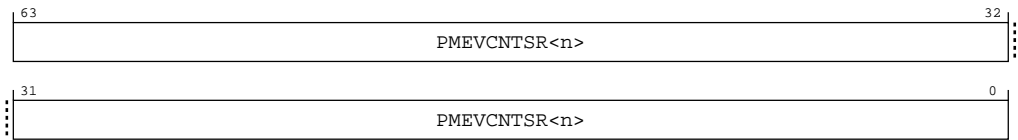


Table B-39: PMEVCNTR8 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

B.3.15 PMEVCNTR9, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

Attributes

Width

64

Functional group

PMU

Register offset

0x668

Reset value

See individual bit resets.

Bit descriptions

Figure B-37: ext\_PMEVCNTR19 bit assignments

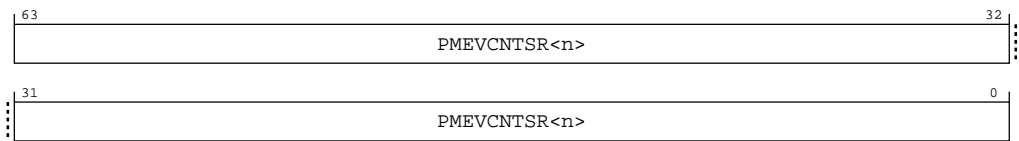


Table B-40: PMEVCNTR9 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

B.3.16 PMEVCNTR10, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

Attributes

Width

64

Functional group

PMU

Register offset

0x670

Reset value

See individual bit resets.

Bit descriptions

Figure B-38: ext\_PMEVCNTR19 bit assignments

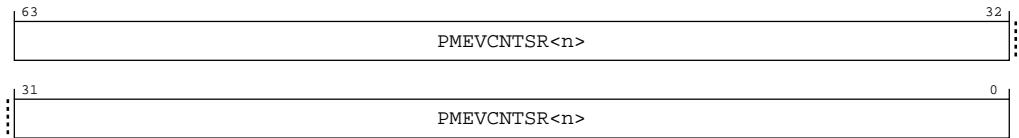


Table B-41: PMEVCNTR10 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

B.3.17 PMEVCNTR11, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

Attributes

Width

64

Functional group

PMU

Register offset

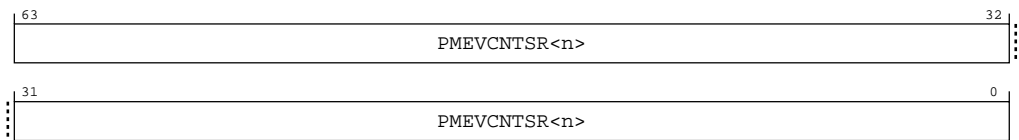
0x678

Reset value

See individual bit resets.

Bit descriptions

Figure B-39: ext\_PMEVCNTR19 bit assignments



**Table B-42: PMEVCNTR11 bit descriptions**

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

### B.3.18 PMEVCNTR12, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

#### Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

#### Attributes

##### Width

64

##### Functional group

PMU

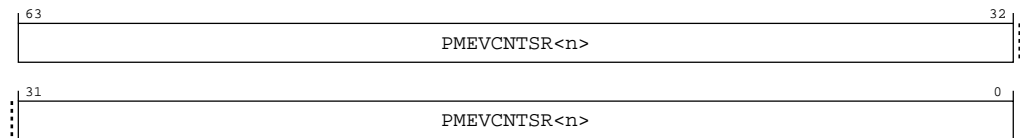
##### Register offset

0x680

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-40: ext\_PMEVCNTR19 bit assignments****Table B-43: PMEVCNTR12 bit descriptions**

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	



### B.3.19 PMEVCNTR13, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

#### Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

#### Attributes

**Width**

64

**Functional group**

PMU

**Register offset**

0x688

**Reset value**

See individual bit resets.

#### Bit descriptions

Figure B-41: ext\_PMEVCNTR19 bit assignments

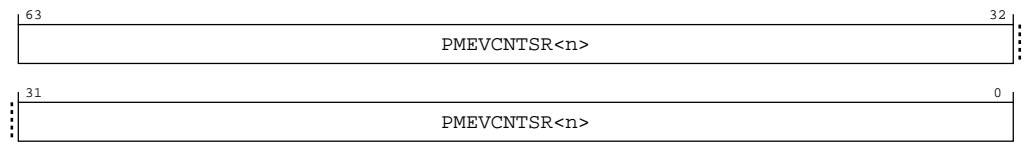


Table B-44: PMEVCNTR13 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

### B.3.20 PMEVCNTR14, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

#### Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

#### Attributes

**Width**

64

Functional group

PMU

Register offset

0x690

Reset value

See individual bit resets.

Bit descriptions

Figure B-42: ext\_PMEVCNTR19 bit assignments

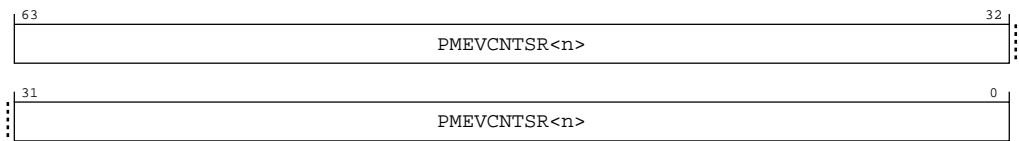


Table B-45: PMEVCNTR14 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

B.3.21 PMEVCNTR15, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

Attributes

Width

64

Functional group

PMU

Register offset

0x698

Reset value

See individual bit resets.

Bit descriptions

Figure B-43: ext\_PMEVCNTR19 bit assignments

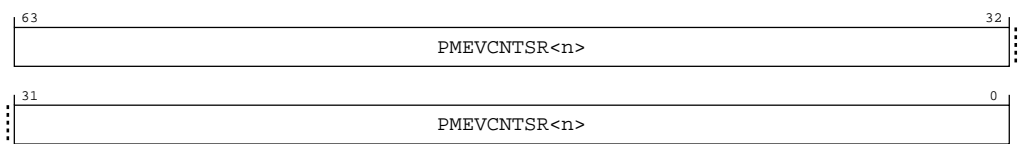


Table B-46: PMEVCNTR15 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

B.3.22 PMEVCNTR16, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

Attributes

Width

64

Functional group

PMU

Register offset

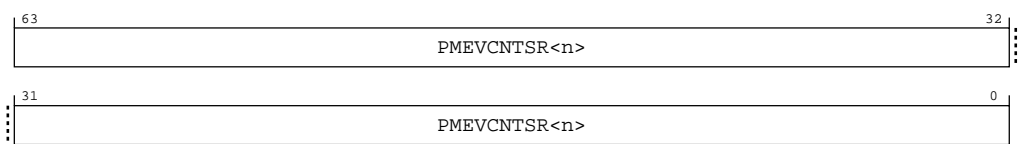
0x6A0

Reset value

See individual bit resets.

Bit descriptions

Figure B-44: ext\_PMEVCNTR19 bit assignments



**Table B-47: PMEVCNTR16 bit descriptions**

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

### B.3.23 PMEVCNTR17, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

#### Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

#### Attributes

##### Width

64

##### Functional group

PMU

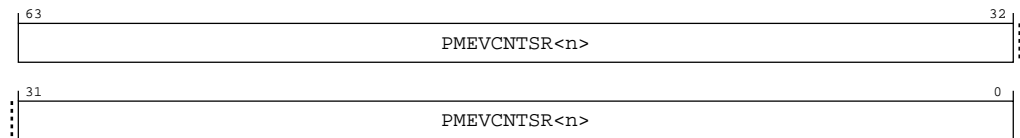
##### Register offset

0x6A8

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-45: ext\_PMEVCNTR19 bit assignments****Table B-48: PMEVCNTR17 bit descriptions**

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

### B.3.24 PMEVCNTR18, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

#### Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

#### Attributes

**Width**

64

**Functional group**

PMU

**Register offset**

0x6B0

**Reset value**

See individual bit resets.

#### Bit descriptions

Figure B-46: ext\_PMEVCNTR19 bit assignments

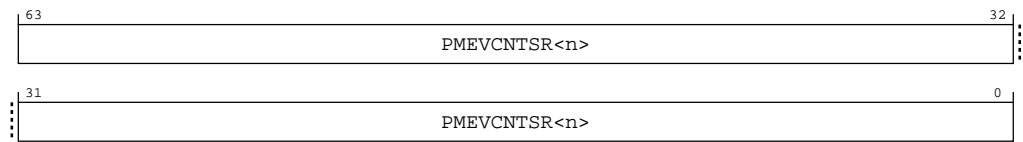


Table B-49: PMEVCNTR18 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

### B.3.25 PMEVCNTR19, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>\_ELO. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>\_ELO and PMCR\_ELO.P.

#### Configurations

- This register is available in configurations where 20 PMU event counters are implemented.

#### Attributes

**Width**

64

Functional group

PMU

Register offset

0x6B8

Reset value

See individual bit resets.

Bit descriptions

Figure B-47: ext\_PMEVCNTR19 bit assignments

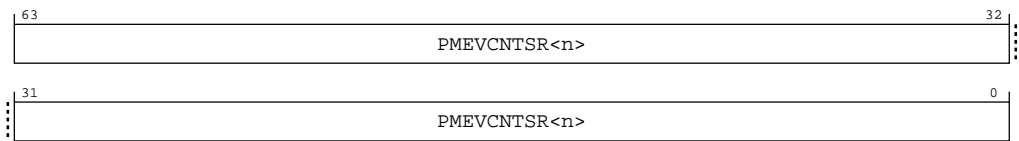


Table B-50: PMEVCNTR19 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTR<n>	PMEVCNTR<n>_ELO sample. Sampled event count.	

B.3.26 PMSSCR, PMU Snapshot Capture Register

Provides a mechanism for software to initiate a sample.

Configurations

- This register is available in all configurations.

Attributes

Width

32

Functional group

PMU

Register offset

0x6F0

Reset value

See individual bit resets.

## Bit descriptions

Figure B-48: ext\_PMSSCR bit assignments

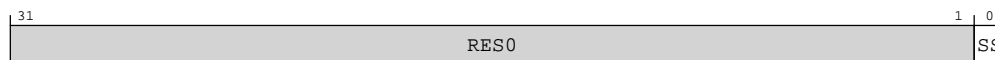


Table B-51: PMSSCR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	0b0
[0]	SS	<p>Capture now.</p> <p><b>0b0</b> Ignored.</p> <p><b>0b1</b> Initiate a capture immediately.</p>	

## B.3.27 PMCFGR, Performance Monitors Configuration Register

Contains PMU-specific configuration data.

## Configurations

- This register is available in all configurations.

## Attributes

## Width

32

## Functional group

PMU

## Register offset

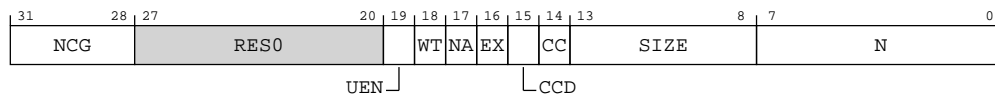
0xE00

## Reset value

See individual bit resets.

## Bit descriptions

Figure B-49: ext\_PMCFG bit assignments



**Table B-52: PMCFGR bit descriptions**

Bits	Name	Description	Reset
[31:28]	NCG	This feature is not supported, so this field is RAZ.	
[27:20]	RESO	Reserved	0b0
[19]	UEN	User-mode Enable Register supported. AArch64-PMUSERENR_ELO is not visible in the external debug interface, so this bit is RAZ.	
[18]	WT	This feature is not supported, so this bit is RAZ.	
[17]	NA	This feature is not supported, so this bit is RAZ.	
[16]	EX	Export supported.  <b>0b1</b> ext-PMCR_ELO.X is read/write.	
[15]	CCD	Cycle counter has prescale.  <b>0b1</b> ext-PMCR_ELO.D is read/write.	
[14]	CC	Dedicated cycle counter (counter 31) supported. This bit is RAO.	
[13:8]	SIZE	Size of counters, minus one. This field defines the size of the largest counter implemented by the Performance Monitors Unit.  From Armv8, the largest counter is 64-bits, so the value of this field is 0b111111.  This field is used by software to determine the spacing of the counters in the memory-map. From Armv8, the counters are a doubleword-aligned addresses.	
[7:0]	N	Number of counters implemented in addition to the cycle counter, ext-PMCCNTR_ELO. The maximum number of event counters is 31.  <b>0b00000000</b> Only ext-PMCCNTR_ELO implemented.  <b>0b00000001</b> ext-PMCCNTR_ELO plus one event counter implemented.	

### B.3.28 PMCR\_ELO, Performance Monitors Control Register

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

PMU



**Register offset**

0xE04

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-50: ext\_PMCR\_ELO bit assignments****Table B-53: PMCR\_ELO bit descriptions**

Bits	Name	Description	Reset
[31:11]	RAZ/WI	Reserved	
[10:8]	RES0	Reserved	0b0
[7]	LP	Long event counter enable. Determines when unsigned overflow is recorded by a counter overflow bit. <b>0b1</b> Event counter overflow on increment that causes unsigned overflow of ext-PMECNTR<n>_ELO[63:0].	
[6]	RES1	Reserved	0b1
[5]	DP	Disable cycle counter when event counting is prohibited. The possible values of this bit are: <b>0b0</b> Cycle counting by ext-PMCCNTR_ELO is not affected by this bit. <b>0b1</b> When event counting for counters in the range [0..(AArch64-MDCR_EL2.HPMN-1)] is prohibited, cycle counting by ext-PMCCNTR_ELO is disabled.	
[4]	X	Enable export of events in an <b>IMPLEMENTATION DEFINED</b> PMU event export bus. <b>0b0</b> Do not export events. <b>0b1</b> Export events where not prohibited.	
[3]	RES0	Reserved	0b0
[2]	C	Cycle counter reset. The effects of writing to this bit are: <b>0b1</b> Reset ext-PMCCNTR_ELO to zero.	
[1]	P	Event counter reset. The effects of writing to this bit are: <b>0b1</b> Reset all event counters, not including ext-PMCCNTR_ELO, to zero.	
[0]	E	Enable <b>0b1</b> All event counters in the range [0..(PMN-1)] and ext-PMCCNTR_ELO, are enabled by ext-PMCNTENSET_ELO.	

### B.3.29 PMCEID0, Performance Monitors Common Event Identification register 0

Defines which common architectural events and common microarchitectural events are implemented, or counted, using PMU events in the range 0x0000 to 0x001F

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

For more information about the common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

- Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0.
- This view of the register was previously called PMCEID0\_ELO.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

PMU

##### Register offset

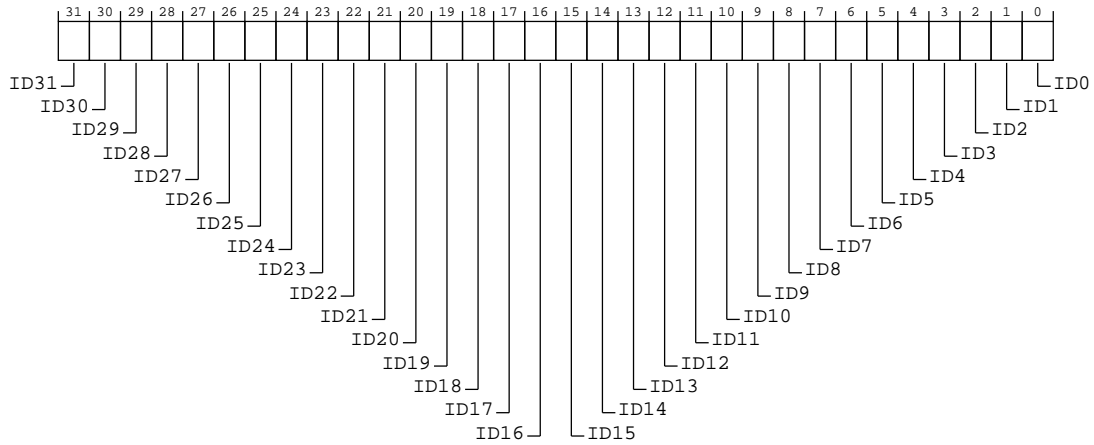
0xE20

##### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-51: ext\_PMCEID0 bit assignments**



**Table B-54: PMCEID0 bit descriptions**

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x1f) L1D_CACHE_ALLOCATE <b>0b0</b> The common event is not implemented, or not counted.	
[30]	ID30	ID30 corresponds to common event (0x1e) CHAIN <b>0b1</b> The common event is implemented.	
[29]	ID29	ID29 corresponds to common event (0x1d) BUS_CYCLES <b>0b1</b> The common event is implemented.	
[28]	ID28	ID28 corresponds to common event (0x1c) TTBR_WRITE_RETIRED <b>0b1</b> The common event is implemented.	
[27]	ID27	ID27 corresponds to common event (0x1b) INST_SPEC <b>0b1</b> The common event is implemented.	
[26]	ID26	ID26 corresponds to common event (0x1a) MEMORY_ERROR <b>0b1</b> The common event is implemented.	
[25]	ID25	ID25 corresponds to common event (0x19) BUS_ACCESS <b>0b1</b> The common event is implemented.	
[24]	ID24	ID24 corresponds to common event (0x18) L2D_CACHE_WB <b>0b1</b> The common event is implemented.	

Bits	Name	Description	Reset
[23]	ID23	ID23 corresponds to common event (0x17) L2D_CACHE_REFILL <b>0b1</b> The common event is implemented.	
[22]	ID22	ID22 corresponds to common event (0x16) L2D_CACHE <b>0b1</b> The common event is implemented.	
[21]	ID21	ID21 corresponds to common event (0x15) L1D_CACHE_WB <b>0b1</b> The common event is implemented.	
[20]	ID20	ID20 corresponds to common event (0x14) L1I_CACHE <b>0b1</b> The common event is implemented.	
[19]	ID19	ID19 corresponds to common event (0x13) MEM_ACCESS <b>0b1</b> The common event is implemented.	
[18]	ID18	ID18 corresponds to common event (0x12) BR_PRED <b>0b1</b> The common event is implemented.	
[17]	ID17	ID17 corresponds to common event (0x11) CPU_CYCLES <b>0b1</b> The common event is implemented.	
[16]	ID16	ID16 corresponds to common event (0x10) BR_MIS_PRED <b>0b1</b> The common event is implemented.	
[15]	ID15	ID15 corresponds to common event (0xf) UNALIGNED_LDST_RETIRED <b>0b0</b> The common event is not implemented, or not counted.	
[14]	ID14	ID14 corresponds to common event (0xe) BR_RETURN_RETIRED <b>0b0</b> The common event is not implemented, or not counted.	
[13]	ID13	ID13 corresponds to common event (0xd) BR_IMMED_RETIRED <b>0b0</b> The common event is not implemented, or not counted.	
[12]	ID12	ID12 corresponds to common event (0xc) PC_WRITE_RETIRED <b>0b0</b> The common event is not implemented, or not counted.	
[11]	ID11	ID11 corresponds to common event (0xb) CID_WRITE_RETIRED <b>0b1</b> The common event is implemented.	
[10]	ID10	ID10 corresponds to common event (0xa) EXC_RETURN <b>0b1</b> The common event is implemented.	

Bits	Name	Description	Reset
[9]	ID9	ID9 corresponds to common event (0x9) EXC_TAKEN <b>0b1</b> The common event is implemented.	
[8]	ID8	ID8 corresponds to common event (0x8) INST_RETIRED <b>0b1</b> The common event is implemented.	
[7]	ID7	ID7 corresponds to common event (0x7) ST_RETIRED <b>0b0</b> The common event is not implemented, or not counted.	
[6]	ID6	ID6 corresponds to common event (0x6) LD_RETIRED <b>0b0</b> The common event is not implemented, or not counted.	
[5]	ID5	ID5 corresponds to common event (0x5) L1D_TLB_REFILL <b>0b1</b> The common event is implemented.	
[4]	ID4	ID4 corresponds to common event (0x4) L1D_CACHE <b>0b1</b> The common event is implemented.	
[3]	ID3	ID3 corresponds to common event (0x3) L1D_CACHE_REFILL <b>0b1</b> The common event is implemented.	
[2]	ID2	ID2 corresponds to common event (0x2) L1I_TLB_REFILL <b>0b1</b> The common event is implemented.	
[1]	ID1	ID1 corresponds to common event (0x1) L1I_CACHE_REFILL <b>0b1</b> The common event is implemented.	
[0]	ID0	ID0 corresponds to common event (0x0) SW_INCR <b>0b1</b> The common event is implemented.	

### B.3.30 PMCEID1, Performance Monitors Common Event Identification register 1

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

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

For more information about the common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

- Arm recommends that, if a common event is never counted, the value of the corresponding register bit is 0.
- This view of the register was previously called PMCEID1\_ELO.

## Configurations

- This register is available in all configurations.

## Attributes

### Width

32

### Functional group

PMU

### Register offset

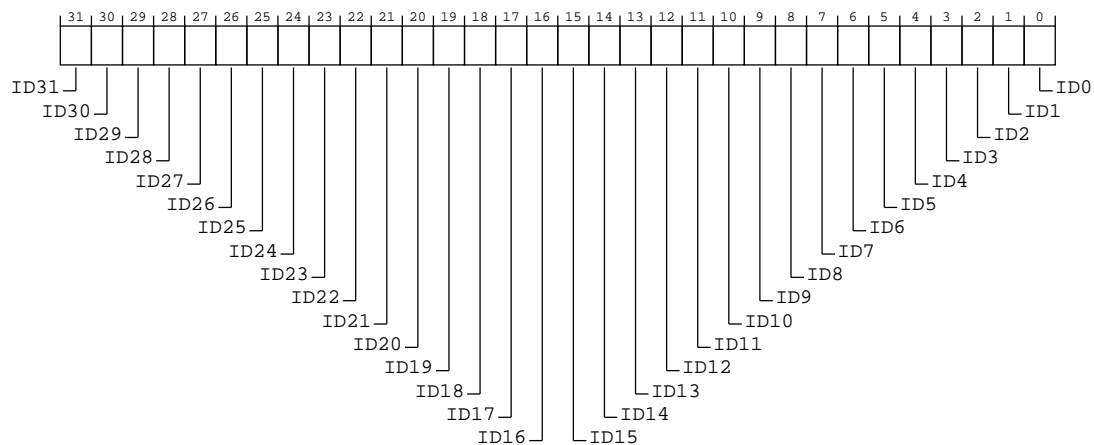
0xE24

### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-52: ext\_PMCEID1 bit assignments**



**Table B-55: PMCEID1 bit descriptions**

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x3f) STALL_SLOT <b>0b1</b> The common event is implemented.	
[30]	ID30	ID30 corresponds to common event (0x3e) STALL_SLOT_FRONTEND <b>0b1</b> The common event is implemented.	

Bits	Name	Description	Reset
[29]	ID29	ID29 corresponds to common event (0x3d) STALL_SLOT_BACKEND <b>0b1</b> The common event is implemented.	
[28]	ID28	ID28 corresponds to common event (0x3c) STALL <b>0b1</b> The common event is implemented.	
[27]	ID27	ID27 corresponds to common event (0x3b) OP_SPEC <b>0b1</b> The common event is implemented.	
[26]	ID26	ID26 corresponds to common event (0x3a) OP_RETIRED <b>0b1</b> The common event is implemented.	
[25]	ID25	ID25 corresponds to common event (0x39) L1D_CACHE_LMISS_RD <b>0b1</b> The common event is implemented.	
[24]	ID24	ID24 corresponds to common event (0x38) REMOTE_ACCESS_RD <b>0b0</b> The common event is not implemented, or not counted.	
[23]	ID23	ID23 corresponds to common event (0x37) LL_CACHE_MISS_RD <b>0b1</b> The common event is implemented.	
[22]	ID22	ID22 corresponds to common event (0x36) LL_CACHE_RD <b>0b1</b> The common event is implemented.	
[21]	ID21	ID21 corresponds to common event (0x35) ITLB_WLK <b>0b1</b> The common event is implemented.	
[20]	ID20	ID20 corresponds to common event (0x34) DTLB_WLK <b>0b1</b> The common event is implemented.	
[19]	ID19	ID19 corresponds to a Reserved Event event (0x33) <b>0b0</b> The common event is not implemented, or not counted.	
[18]	ID18	ID18 corresponds to a Reserved Event event (0x32) <b>0b0</b> The common event is not implemented, or not counted.	
[17]	ID17	ID17 corresponds to common event (0x31) REMOTE_ACCESS <b>0b1</b> The common event is implemented.	
[16]	ID16	ID16 corresponds to common event (0x30) L2I_TLB <b>0b0</b> The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[15]	ID15	ID15 corresponds to common event (0x2f) L2TLB_REQ <b>0b1</b> The common event is implemented.	
[14]	ID14	ID14 corresponds to common event (0x2e) L2I_TLB_REFILL <b>0b0</b> The common event is not implemented, or not counted.	
[13]	ID13	ID13 corresponds to common event (0x2d) L2TLB_REFILL <b>0b1</b> The common event is implemented.	
[12]	ID12	ID12 corresponds to common event (0x2c) Reserved <b>0b0</b> The common event is not implemented, or not counted.	
[11]	ID11	ID11 corresponds to common event (0x2b) L3D_CACHE <b>0b1</b> The common event is implemented.	
[10]	ID10	ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL <b>0b1</b> The common event is implemented.	
[9]	ID9	ID9 corresponds to common event (0x29) L3D_CACHE_ALLOCATE <b>0b1</b> The common event is implemented.	
[8]	ID8	ID8 corresponds to common event (0x28) L2I_CACHE_REFILL <b>0b0</b> The common event is not implemented, or not counted.	
[7]	ID7	ID7 corresponds to common event (0x27) L2I_CACHE <b>0b0</b> The common event is not implemented, or not counted.	
[6]	ID6	ID6 corresponds to common event (0x26) L1I_TLB <b>0b1</b> The common event is implemented.	
[5]	ID5	ID5 corresponds to common event (0x25) L1D_TLB <b>0b1</b> The common event is implemented.	
[4]	ID4	ID4 corresponds to common event (0x24) STALL_BACKEND <b>0b1</b> The common event is implemented.	
[3]	ID3	ID3 corresponds to common event (0x23) STALL_FRONTEND <b>0b1</b> The common event is implemented.	
[2]	ID2	ID2 corresponds to common event (0x22) BR_MIS_PRED_RETIRED <b>0b1</b> The common event is implemented.	



Bits	Name	Description	Reset
[1]	ID1	ID1 corresponds to common event (0x21) BR_RETIRED  <b>0b1</b> The common event is implemented.	
[0]	ID0	ID0 corresponds to common event (0x20) L2D_CACHE_ALLOCATE  <b>0b1</b> The common event is implemented.	

### B.3.31 PMCEID2, Performance Monitors Common Event Identification register 2

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

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



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

For more information about the common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

PMU

##### Register offset

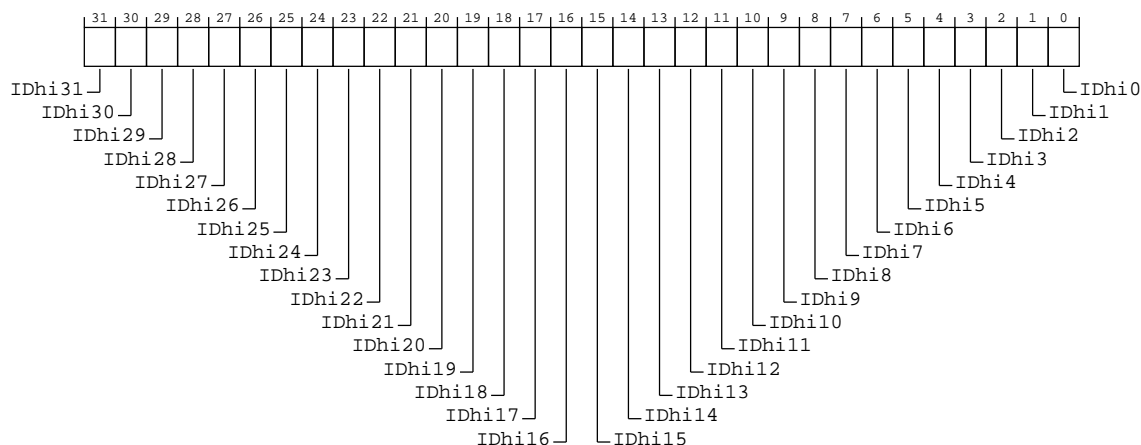
0xE28

##### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-53: ext\_PMCEID2 bit assignments**



**Table B-56: PMCEID2 bit descriptions**

Bits	Name	Description	Reset
[31]	IDHi31	IDHi31 corresponds to a Reserved Event event (0x401f) <b>0b0</b> The common event is not implemented, or not counted.	
[30]	IDHi30	IDHi30 corresponds to a Reserved Event event (0x401e) <b>0b0</b> The common event is not implemented, or not counted.	
[29]	IDHi29	IDHi29 corresponds to a Reserved Event event (0x401d) <b>0b0</b> The common event is not implemented, or not counted.	
[28]	IDHi28	IDHi28 corresponds to a Reserved Event event (0x401c) <b>0b0</b> The common event is not implemented, or not counted.	
[27]	IDHi27	IDHi27 corresponds to common event (0x401b) CTI_TRIGOUT7 <b>0b1</b> The common event is implemented.	
[26]	IDHi26	IDHi26 corresponds to common event (0x401a) CTI_TRIGOUT6 <b>0b1</b> The common event is implemented.	
[25]	IDHi25	IDHi25 corresponds to common event (0x4019) CTI_TRIGOUT5 <b>0b1</b> The common event is implemented.	
[24]	IDHi24	IDHi24 corresponds to common event (0x4018) CTI_TRIGOUT4 <b>0b1</b> The common event is implemented.	

Bits	Name	Description	Reset
[23]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4017) <b>0b0</b> The common event is not implemented, or not counted.	
[22]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4016) <b>0b0</b> The common event is not implemented, or not counted.	
[21]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4015) <b>0b0</b> The common event is not implemented, or not counted.	
[20]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4014) <b>0b0</b> The common event is not implemented, or not counted.	
[19]	IDhi19	IDhi19 corresponds to common event (0x4013) TRCEXTOUT3 <b>0b1</b> The common event is implemented.	
[18]	IDhi18	IDhi18 corresponds to common event (0x4012) TRCEXTOUT2 <b>0b1</b> The common event is implemented.	
[17]	IDhi17	IDhi17 corresponds to common event (0x4011) TRCEXTOUT1 <b>0b1</b> The common event is implemented.	
[16]	IDhi16	IDhi16 corresponds to common event (0x4010) TRCEXTOUT0 <b>0b1</b> The common event is implemented.	
[15]	IDhi15	IDhi15 corresponds to common event (0x400f) Reserved <b>0b0</b> The common event is not implemented, or not counted.	
[14]	IDhi14	IDhi14 corresponds to common event (0x400e) TRB_TRIG <b>0b0</b> The common event is not implemented, or not counted.	
[13]	IDhi13	IDhi13 corresponds to common event (0x400d) PMU_OVFS <b>0b0</b> The common event is not implemented, or not counted.	
[12]	IDhi12	IDhi12 corresponds to common event (0x400c) TRB_WRAP <b>0b1</b> The common event is implemented.	
[11]	IDhi11	IDhi11 corresponds to common event (0x400b) L3D_CACHE_LMISS_RD <b>0b1</b> The common event is implemented.	
[10]	IDhi10	IDhi10 corresponds to common event (0x400a) L2I_CACHE_LMISS <b>0b0</b> The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[9]	IDhi9	IDhi9 corresponds to common event (0x4009) L2D_CACHE_LMISS_RD <b>0b1</b> The common event is implemented.	
[8]	IDhi8	IDhi8 corresponds to common event (0x4008) Reserved <b>0b0</b> The common event is not implemented, or not counted.	
[7]	IDhi7	IDhi7 corresponds to common event (0x4007) Reserved <b>0b0</b> The common event is not implemented, or not counted.	
[6]	IDhi6	IDhi6 corresponds to common event (0x4006) L1I_CACHE_LMISS <b>0b1</b> The common event is implemented.	
[5]	IDhi5	IDhi5 corresponds to common event (0x4005) STALL_BACKEND_MEM <b>0b1</b> The common event is implemented.	
[4]	IDhi4	IDhi4 corresponds to common event (0x4004) CNT_CYCLES <b>0b1</b> The common event is implemented.	
[3]	IDhi3	IDhi3 corresponds to common event (0x4003) SAMPLE_COLLISION <b>0b0</b> The common event is not implemented, or not counted.	
[2]	IDhi2	IDhi2 corresponds to common event (0x4002) SAMPLE_FILTRATE <b>0b0</b> The common event is not implemented, or not counted.	
[1]	IDhi1	IDhi1 corresponds to common event (0x4001) SAMPLE_FEED <b>0b0</b> The common event is not implemented, or not counted.	
[0]	IDhi0	IDhi0 corresponds to common event (0x4000) SAMPLE_POP <b>0b0</b> The common event is not implemented, or not counted.	

### B.3.32 PMCEID3, Performance Monitors Common Event Identification register 3

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

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



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

For more information about the common events and the use of the PMCEIDn registers, see 'The PMU event number space and common events'.

## Configurations

- This register is available in all configurations.

## Attributes

### Width

32

### Functional group

PMU

### Register offset

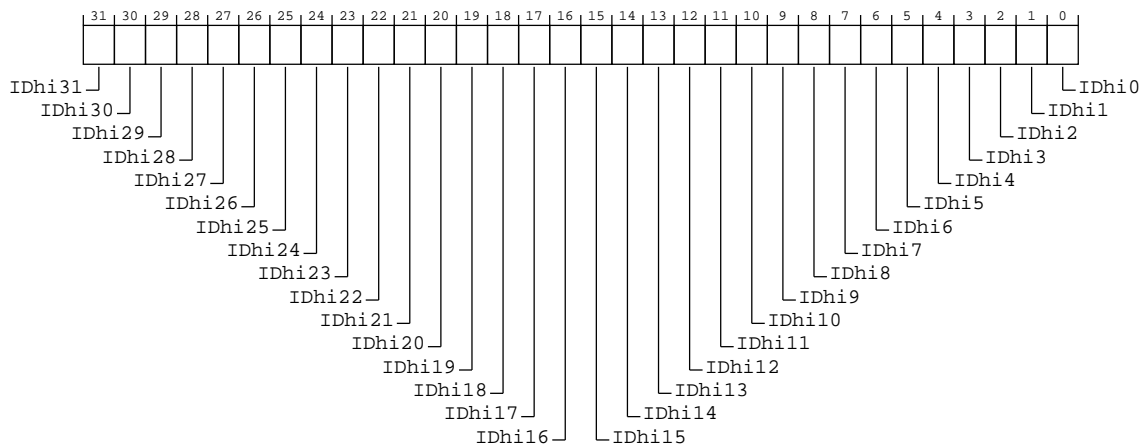
0xE2C

### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-54: ext\_PMCEID3 bit assignments**



**Table B-57: PMCEID3 bit descriptions**

Bits	Name	Description	Reset
[31]	IDHi31	IDHi31 corresponds to a Reserved Event event (0x403f)  <b>0b0</b> The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[30]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x403e) <b>0b0</b> The common event is not implemented, or not counted.	
[29]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x403d) <b>0b0</b> The common event is not implemented, or not counted.	
[28]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x403c) <b>0b0</b> The common event is not implemented, or not counted.	
[27]	IDhi27	IDhi27 corresponds to a Reserved Event event (0x403b) <b>0b0</b> The common event is not implemented, or not counted.	
[26]	IDhi26	IDhi26 corresponds to a Reserved Event event (0x403a) <b>0b0</b> The common event is not implemented, or not counted.	
[25]	IDhi25	IDhi25 corresponds to a Reserved Event event (0x4039) <b>0b0</b> The common event is not implemented, or not counted.	
[24]	IDhi24	IDhi24 corresponds to a Reserved Event event (0x4038) <b>0b0</b> The common event is not implemented, or not counted.	
[23]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4037) <b>0b0</b> The common event is not implemented, or not counted.	
[22]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4036) <b>0b0</b> The common event is not implemented, or not counted.	
[21]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4035) <b>0b0</b> The common event is not implemented, or not counted.	
[20]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4034) <b>0b0</b> The common event is not implemented, or not counted.	
[19]	IDhi19	IDhi19 corresponds to a Reserved Event event (0x4033) <b>0b0</b> The common event is not implemented, or not counted.	
[18]	IDhi18	IDhi18 corresponds to a Reserved Event event (0x4032) <b>0b0</b> The common event is not implemented, or not counted.	
[17]	IDhi17	IDhi17 corresponds to a Reserved Event event (0x4031) <b>0b0</b> The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[16]	IDhi16	IDhi16 corresponds to a Reserved Event event (0x4030) <b>0b0</b> The common event is not implemented, or not counted.	
[15]	IDhi15	IDhi15 corresponds to a Reserved Event event (0x402f) <b>0b0</b> The common event is not implemented, or not counted.	
[14]	IDhi14	IDhi14 corresponds to a Reserved Event event (0x402e) <b>0b0</b> The common event is not implemented, or not counted.	
[13]	IDhi13	IDhi13 corresponds to a Reserved Event event (0x402d) <b>0b0</b> The common event is not implemented, or not counted.	
[12]	IDhi12	IDhi12 corresponds to a Reserved Event event (0x402c) <b>0b0</b> The common event is not implemented, or not counted.	
[11]	IDhi11	IDhi11 corresponds to a Reserved Event event (0x402b) <b>0b0</b> The common event is not implemented, or not counted.	
[10]	IDhi10	IDhi10 corresponds to a Reserved Event event (0x402a) <b>0b0</b> The common event is not implemented, or not counted.	
[9]	IDhi9	IDhi9 corresponds to a Reserved Event event (0x4029) <b>0b0</b> The common event is not implemented, or not counted.	
[8]	IDhi8	IDhi8 corresponds to a Reserved Event event (0x4028) <b>0b0</b> The common event is not implemented, or not counted.	
[7]	IDhi7	IDhi7 corresponds to a Reserved Event event (0x4027) <b>0b0</b> The common event is not implemented, or not counted.	
[6]	IDhi6	IDhi6 corresponds to common event (0x4026) MEM_ACCESS_CHECKED_WR <b>0b1</b> The common event is implemented.	
[5]	IDhi5	IDhi5 corresponds to common event (0x4025) MEM_ACCESS_CHECKED_RD <b>0b1</b> The common event is implemented.	
[4]	IDhi4	IDhi4 corresponds to common event (0x4024) MEM_ACCESS_CHECKED <b>0b1</b> The common event is implemented.	
[3]	IDhi3	IDhi3 corresponds to common event (0x4023) Reserved <b>0b0</b> The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[2]	IDhi2	IDhi2 corresponds to common event (0x4022) ST_ALIGN_LAT  <b>0b1</b> The common event is implemented.	
[1]	IDhi1	IDhi1 corresponds to common event (0x4021) LD_ALIGN_LAT  <b>0b1</b> The common event is implemented.	
[0]	IDhi0	IDhi0 corresponds to common event (0x4020) LDST_ALIGN_LAT  <b>0b1</b> The common event is implemented.	

### B.3.33 PMMIR, Performance Monitors Machine Identification Register

Describes Performance Monitors parameters specific to the implementation.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

PMU

##### Register offset

0xE40

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-55: ext\_PMMIR bit assignments**



**Table B-58: PMMIR bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	SLOTS	Operation width. The largest value by which the STALL_SLOT event might increment by in a single cycle. If the STALL_SLOT event is implemented, this field must not be zero.	



### B.3.34 PMDEVARCH, Performance Monitors Device Architecture register

Identifies the programmers' model architecture of the Performance Monitor component.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

PMU

##### Register offset

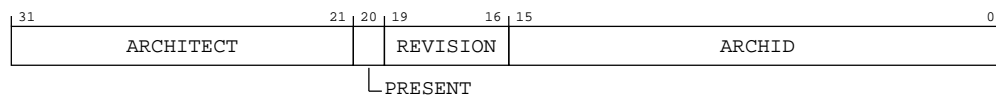
0xFBC

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-56: ext\_PMDEVARCH bit assignments**



**Table B-59: PMDEVARCH bit descriptions**

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For Performance Monitors, this is Arm Limited.  Bits [31:28] are the JEP106 continuation code, 0x4.  Bits [27:21] are the JEP106 ID code, 0x3B.	
[20]	PRESENT	When set to 1, indicates that the DEVARCH is present.  This field is 1 in Armv8.	
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision.  For Performance Monitors, the revision defined by Armv8 is 0x0.  All other values are reserved.	

Bits	Name	Description	Reset
[15:0]	ARCHID	<p>Defines this part to be an Armv8 debug component. For architectures defined by Arm this is further subdivided.</p> <p>For Performance Monitors:</p> <ul style="list-style-type: none"> <li>Bits [15:12] are the architecture version, 0x2.</li> <li>Bits [11:0] are the architecture part number, 0xA16. This corresponds to Performance Monitors architecture version PMUv3.</li> </ul> <p><b>Note:</b> 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 ext-PMDEVAFF0 and ext-PMDEVAFF1 registers to discover the affinity of the PMU to any Armv8 processors.</p>	

### B.3.35 PMDEVID, Performance Monitors Device ID register

Provides information about features of the Performance Monitors implementation.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

PMU

##### Register offset

0xFC8

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-57: ext\_PMDEVID bit assignments**



**Table B-60: PMDEVID bit descriptions**

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using Performance Monitors registers.  <b>0b0001</b> PC Sample-based Profiling Extension is implemented in the Performance Monitors register space.	

### B.3.36 PMDEVTYPE, Performance Monitors Device Type register

Indicates to a debugger that this component is part of a PE's performance monitor interface.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

PMU

##### Register offset

0xFCC

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-58: ext\_PMDEVTYPE bit assignments**



**Table B-61: PMDEVTYPE bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	SUB	Subtype. Must read as 0x1 to indicate this is a component within a PE.	
[3:0]	MAJOR	Major type. Must read as 0x6 to indicate this is a performance monitor component.	

### B.3.37 PMPIDR4, Performance Monitors Peripheral Identification Register 4

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

- This register is available in all configurations.

Attributes

Width

32

Functional group

PMU

Register offset

0xFD0

Reset value

See individual bit resets.

Bit descriptions

Figure B-59: ext\_PMPIDR4 bit assignments



Table B-62: PMPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	SIZE	4KB count.  0b0000 The component uses a single 4KB block.	
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100.  0b0100 Arm Limited. This is bits[3:0] of the JEP106 continuation code.	

B.3.38 PMPIDR0, Performance Monitors Peripheral Identification Register 0

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

PMU

**Register offset**

0xFE0

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-60: ext\_PMPIDR0 bit assignments****Table B-63: PMPIDR0 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PART_0	Part number, least significant byte. <b>0b01001000</b> Least significant byte of the PMU unit part.	

### B.3.39 PMPIDR1, Performance Monitors Peripheral Identification Register 1

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

PMU

**Register offset**

0xFE4

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-61: ext\_PMPIDR1 bit assignments****Table B-64: PMPIDR1 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is 0b1011.  <b>0b1011</b> Arm Limited. This is the least significant nibble of JEP106 ID code.	
[3:0]	PART_1	Part number, most significant nibble.  <b>0b1101</b> Part number, most significant nibble.	

## B.3.40 PMPIDR2, Performance Monitors Peripheral Identification Register 2

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

PMU

**Register offset**

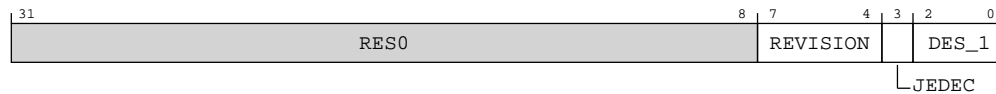
0xFE8

**Reset value**

See individual bit resets.

## Bit descriptions

**Figure B-62: ext\_PMPIDR2 bit assignments**



**Table B-65: PMPIDR2 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits.  <b>0b0010</b> r2p1	
[3]	JEDEC	RAO. Indicates a JEP106 identity code is used.  <b>0b1</b> RES1. Indicates a JEP106 identity code is used	
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011.  <b>0b011</b> Arm Limited. This is bits[6:4] of the JEP106 ID code.	

## B.3.41 PMPIDR3, Performance Monitors Peripheral Identification Register 3

Provides information to identify a Performance Monitor component.

For more information, see 'About the Peripheral identification scheme'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

PMU

#### Register offset

0xFEC

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-63: ext\_PMPIDR3 bit assignments**



**Table B-66: PMPIDR3 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	REVAND	Part minor revision. Parts using ext-PMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. <b>0b0001</b>	
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. <b>0b0000</b> The component is not modified from the original design.	

## B.3.42 PMCIDR0, Performance Monitors Component Identification Register 0

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

PMU

#### Register offset

0xFF0

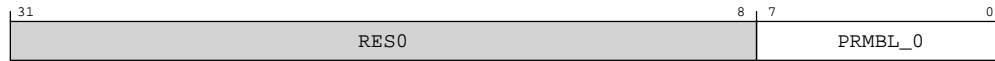
#### Reset value

See individual bit resets.



## Bit descriptions

**Figure B-64: ext\_PMCIDR0 bit assignments**



**Table B-67: PMCIDR0 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_0	Preamble. Must read as 0x0D. <b>0b00001101</b> Preamble byte 0	

## B.3.43 PMCIDR1, Performance Monitors Component Identification Register 1

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

PMU

#### Register offset

0xFF4

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-65: ext\_PMCIDR1 bit assignments**



**Table B-68: PMCIDR1 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	CLASS	Component class. Reads as 0x9, debug component. <b>0b1001</b> Debug Component	
[3:0]	PRMBL_1	Preamble. RAZ. <b>0b0000</b> Preamble	

### B.3.44 PMCIDR2, Performance Monitors Component Identification Register 2

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

PMU

##### Register offset

0xFF8

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-66: ext\_PMCIDR2 bit assignments****Table B-69: PMCIDR2 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[7:0]	PRMBL_2	Preamble. Must read as 0x05.  <b>0b00000101</b> Preamble byte 2.	

### B.3.45 PMCIDR3, Performance Monitors Component Identification Register 3

Provides information to identify a Performance Monitor component.

For more information, see 'About the Component Identification scheme'.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

PMU

##### Register offset

0xFFC

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-67: ext\_PMCIDR3 bit assignments**



**Table B-70: PMCIDR3 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_3	Preamble. Must read as 0xB1.  <b>0b10110001</b> Preamble byte 3.	

## B.4 Debug register summary

The summary table provides an overview of all Debug registers in the core. Individual register descriptions provide detailed information.

**Table B-71: Debug register summary**

Offset	Name	Reset	Width	Description
0x090	EDRCR	See individual bit resets.	32-bit	External Debug Reserve Control Register
0x094	EDACR	0x0	32-bit	External Debug Auxiliary Control Register
0x310	EDPRCR	See individual bit resets.	32-bit	External Debug Power/Reset Control Register
0xD00	MIDR_EL1	See individual bit resets.	32-bit	Main ID Register
0xD20	EDPFR	See individual bit resets.	64-bit	External Debug Processor Feature Register
0xD28	EDDFR	See individual bit resets.	64-bit	External Debug Feature Register
0xFBC	EDDEVARCH	See individual bit resets.	32-bit	External Debug Device Architecture register
0xFC0	EDDEVID2	0x0	32-bit	External Debug Device ID register 2
0xFC4	EDDEVID1	See individual bit resets.	32-bit	External Debug Device ID register 1
0xFC8	EDDEVID	See individual bit resets.	32-bit	External Debug Device ID register 0
0xFCC	EDDEVTYPE	See individual bit resets.	32-bit	External Debug Device Type register
0xFD0	EDPIDR4	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 4
0xFE0	EDPIDR0	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 0
0xFE4	EDPIDR1	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 1
0xFE8	EDPIDR2	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 2
0xFEC	EDPIDR3	See individual bit resets.	32-bit	External Debug Peripheral Identification Register 3
0xFF0	EDCIDR0	See individual bit resets.	32-bit	External Debug Component Identification Register 0
0xFF4	EDCIDR1	See individual bit resets.	32-bit	External Debug Component Identification Register 1
0xFF8	EDCIDR2	See individual bit resets.	32-bit	External Debug Component Identification Register 2
0xFFC	EDCIDR3	See individual bit resets.	32-bit	External Debug Component Identification Register 3

### B.4.1 EDRCR, External Debug Reserve Control Register

This register is used to allow imprecise entry to Debug state and clear sticky bits in ext-EDSCR.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

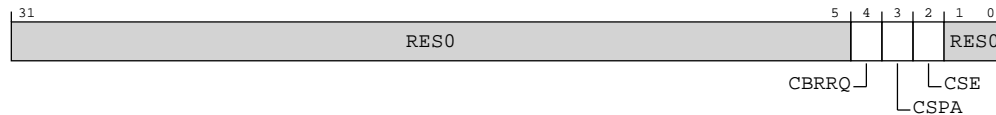
Debug

**Register offset**

0x090

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-68: ext\_EDRCR bit assignments****Table B-72: EDRCR bit descriptions**

Bits	Name	Description	Reset
[31:5]	RES0	Reserved	0b0
[4]	CBRRQ	This feature is not supported. Writes to this bit are ignored.  <b>0b0</b> No action.  <b>0b1</b> Allow imprecise entry to Debug state, for example by canceling pending bus accesses.	
[3]	CSPA	Clear Sticky Pipeline Advance. This bit is used to clear the ext-EDSCR.PipeAdv bit to 0.  <b>0b0</b> No action.  <b>0b1</b> Clear the ext-EDSCR.PipeAdv bit to 0.	
[2]	CSE	Clear Sticky Error. Used to clear the ext-EDSCR cumulative error bits to 0.  <b>0b0</b> No action.  <b>0b1</b> Clear the ext-EDSCR.{TXU, RXO, ERR} bits, and, if the PE is in Debug state, the ext-EDSCR.ITO bit, to 0.	
[1:0]	RES0	Reserved	0b0

**B.4.2 EDACR, External Debug Auxiliary Control Register**Allows implementations to support **IMPLEMENTATION DEFINED** controls.**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

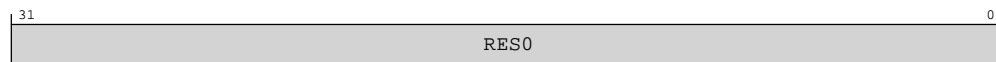
Debug

**Register offset**

0x094

**Reset value**

0x0

**Bit descriptions****Figure B-69: ext\_EDACR bit assignments****Table B-73: EDACR bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

### B.4.3 EDPRCR, External Debug Power/Reset Control Register

Controls the PE functionality related to powerup, reset, and powerdown.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

Debug

**Register offset**

0x310

**Reset value**

See individual bit resets.

Bit descriptions

Figure B-70: ext\_EDPRCR bit assignments

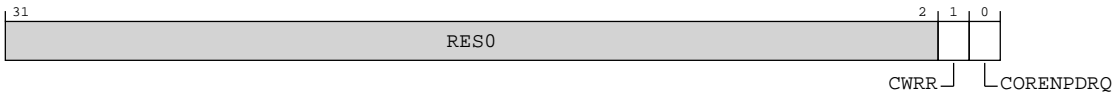


Table B-74: EDPRCR bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	0b0
[1]	CWRR	This feature is not supported. Writes to this bit are ignored  <b>0b0</b> No action.  <b>0b1</b> Request Warm reset.	
[0]	CORENPDRQ	This field is in the Core power domain, and permitted accesses to this field map to the AArch32-DBGPRCR.CORENPDRQ and AArch64-DBGPRCR_EL1.CORENPDRQ fields.  <b>0b0</b> If the system responds to a powerdown request, it powers down Core power domain.  <b>0b1</b> If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain.	

B.4.4 MIDR\_EL1, Main ID Register

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

Configurations

- This register is available in all configurations.

Attributes

Width

32

Functional group

Debug

Register offset

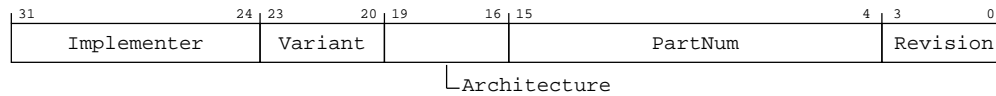
0xD00

Reset value

See individual bit resets.

## Bit descriptions

**Figure B-71: ext\_MIDR\_EL1 bit assignments**



**Table B-75: MIDR\_EL1 bit descriptions**

Bits	Name	Description	Reset
[31:24]	Implementer	Indicates the implementer code. This value is:  <b>0b01000001</b> Arm Limited	
[23:20]	Variant	An <b>IMPLEMENTATION DEFINED</b> variant number. Typically, this field is used to distinguish between different product variants, or major revisions of a product.  <b>0b0010</b> r2p1	
[19:16]	Architecture	Indicates the architecture code. This value is:  <b>0b1111</b> Architecture is defined by ID registers	
[15:4]	PartNum	An <b>IMPLEMENTATION DEFINED</b> 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.  <b>0b110101001000</b> Cortex-X2	
[3:0]	Revision	An <b>IMPLEMENTATION DEFINED</b> revision number for the device.  <b>0b0001</b> r2p1	

## B.4.5 EDPFR, External Debug Processor Feature Register

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'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

64



**Functional group**

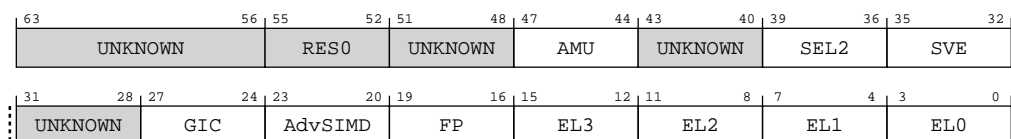
Debug

**Register offset**

0xD20

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-72: ext\_EDPFR bit assignments****Table B-76: EDPFR bit descriptions**

Bits	Name	Description	Reset
[63:56]	UNKNOWN	Reserved	
[55:52]	RES0	Reserved	0b0
[51:48]	UNKNOWN	Reserved	
[47:44]	AMU	Activity Monitors Extension. This value is : <b>0b0001</b> Activity Monitors Extension version 1 is implemented.	
[43:40]	UNKNOWN	Reserved	
[39:36]	SEL2	Secure EL2. This value is : <b>0b0001</b> Secure EL2 is implemented.	
[35:32]	SVE	Scalable Vector Extension. This value is : <b>0b0001</b> SVE is implemented.	
[31:28]	UNKNOWN	Reserved	
[27:24]	GIC	System register GIC interface support <b>0b0011</b> System register interface to version 4.1 of the GIC CPU interface is supported.	
[23:20]	AdvSIMD	Advanced SIMD. This value is: <b>0b0001</b> As for 0b0000, and also includes support for half-precision floating-point arithmetic.	
[19:16]	FP	Floating Point. This value is: <b>0b0001</b> As for 0b0000, and also includes support for half-precision floating-point arithmetic.	

Bits	Name	Description	Reset
[15:12]	EL3	AArch64 EL3 Exception level handling <b>0b0001</b> EL3 can be executed in AArch64 state only.	
[11:8]	EL2	AArch64 EL2 Exception level handling <b>0b0001</b> EL2 can be executed in AArch64 state only.	
[7:4]	EL1	AArch64 EL1 Exception level handling <b>0b0001</b> EL1 can be executed in AArch64 state only.	
[3:0]	EL0	AArch64 EL0 Exception level handling <b>0b0010</b> EL0 can be executed in both Execution states.	

## B.4.6 EDDFR, External Debug Feature Register

Provides top level information about the debug system.



Debuggers must use ext-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'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

64

#### Functional group

Debug

#### Register offset

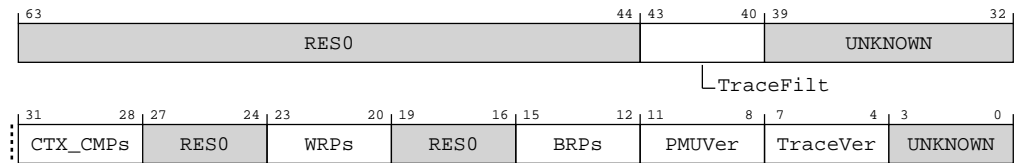
0xD28

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-73: ext\_EDDFR bit assignments**



**Table B-77: EDDFR bit descriptions**

Bits	Name	Description	Reset
[63:44]	RES0	Reserved	0b0
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. This value is : <b>0b0001</b> Armv8.4 Self-hosted Trace Extension is implemented.	
[39:32]	UNKNOWN	Reserved	
[31:28]	CTX_CMPs	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 AArch64-ID_AA64DFR0_EL1.CTX_CMPs.	
[27:24]	RES0	Reserved	0b0
[23:20]	WRPs	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 AArch64-ID_AA64DFR0_EL1.WRPs.	
[19:16]	RES0	Reserved	0b0
[15:12]	BRPs	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 AArch64-ID_AA64DFR0_EL1.BRPs.	
[11:8]	PMUVer	Performance Monitors Extension version. <b>0b0110</b> PMUv3 for Armv8.5. As 0b0101, and also includes support for:  64-bit event counters.  If EL2 is implemented, the AArch64-MDCR_EL2.HCCD control bit.  If EL3 is implemented, the AArch64-MDCR_EL3.SCCD control bit.	
[7:4]	TraceVer	Trace support. Indicates whether System register interface to a PE trace unit is implemented. <b>0b0001</b> PE trace unit System registers implemented.	
[3:0]	UNKNOWN	Reserved	

### B.4.7 EDDEVARCH, External Debug Device Architecture register

Identifies the programmers' model architecture of the external debug component.

## Configurations

- This register is available in all configurations.

## Attributes

## Width

32

### Functional group

Debug

## Register offset

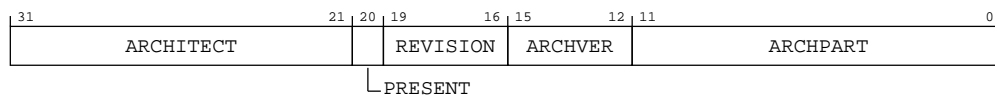
0xFBC

## Reset value

See individual bit resets.

## Bit descriptions

### Figure B-74: ext\_EDDEVARCH bit assignments



### Table B-78: EDDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	<p>Defines the architecture of the component. For debug, this is Arm Limited.</p> <p>Bits [31:28] are the JEP106 continuation code, 0x4.</p> <p>Bits [27:21] are the JEP106 ID code, 0x3B.</p>	
[20]	PRESENT	<p>When set to 1, indicates that the DEVARCH is present.</p> <p>This field is 1 in Armv8.</p>	
[19:16]	REVISION	<p>Defines the architecture revision. For architectures defined by Arm this is the minor revision.</p> <p>For debug, the revision defined by Armv8-A is 0x0.</p> <p>All other values are reserved.</p>	
[15:12]	ARCHVER	<p>Defines the architecture version of the component. This is the same value as AArch64-ID_AA64DFRO_EL1.DebugVer and AArch32-DBGDIDR.Version. This value is :</p> <p><b>0b1001</b></p> <p>Armv8.4 Debug architecture.</p>	

Bits	Name	Description	Reset
[11:0]	ARCHPART	The fields ARCHVER and ARCHPART together form the field ARCHID, so that ARCHPART is ARCHID[11:0]. <b>0b1010000010101</b> The part number of the Armv8-A debug component.	

## B.4.8 EDDEVID2, External Debug Device ID register 2

Reserved for future descriptions of features of the debug implementation.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

Debug

#### Register offset

0xFC0

#### Reset value

0x0

### Bit descriptions

**Figure B-75: ext\_EDDEVID2 bit assignments**



**Table B-79: EDDEVID2 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

## B.4.9 EDDEVID1, External Debug Device ID register 1

Provides extra information for external debuggers about features of the debug implementation.

### Configurations

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

Debug

**Register offset**

0xFC4

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-76: ext\_EDDEVID1 bit assignments****Table B-80: EDDEVID1 bit descriptions**

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	0b0
[3:0]	PCSROffset	This field indicates the offset applied to PC samples returned by reads of ext-EDPCSR.  <b>0b0000</b> ext-EDPCSR not implemented.	

**B.4.10 EDDEVID, External Debug Device ID register 0**

Provides extra information for external debuggers about features of the debug implementation.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

Debug

**Register offset**

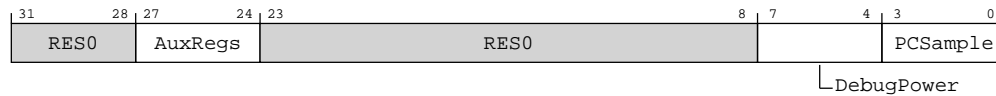
0xFC8

**Reset value**

See individual bit resets.

## Bit descriptions

**Figure B-77: ext\_EDDEVID bit assignments**



**Table B-81: EDDEVID bit descriptions**

Bits	Name	Description	Reset
[31:28]	RES0	Reserved	0b0
[27:24]	AuxRegs	Indicates support for Auxiliary registers.  <b>0b0000</b> None supported.	
[23:8]	RES0	Reserved	0b0
[7:4]	DebugPower	Indicates support for the ARMv8.3-DoPD feature.  <b>0b0001</b> ARMv8.3-DoPD implemented. All registers in the external debug interface register map are implemented in the Core power domain.	
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using external debug registers.  <b>0b0000</b> PC Sample-based Profiling Extension is not implemented in the external debug registers space.	

### B.4.11 EDDEVTYPE, External Debug Device Type register

Indicates to a debugger that this component is part of a PE's debug logic.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

Debug

##### Register offset

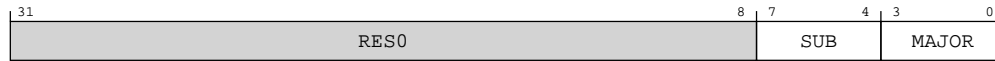
0xFCC

##### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-78: ext\_EDDEVTYPE bit assignments**



**Table B-82: EDDEVTYPE bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	SUB	Subtype. Must read as 0x1 to indicate this is a component within a PE.	
[3:0]	MAJOR	Major type. Must read as 0x5 to indicate this is a debug logic component.	

## B.4.12 EDPIDR4, External Debug Peripheral Identification Register 4

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

Debug

#### Register offset

0xFD0

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-79: ext\_EDPIDR4 bit assignments**



**Table B-83: EDPIDR4 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0



Bits	Name	Description	Reset
[7:4]	SIZE	<p>4KB count.</p> <p><b>0b0000</b></p> <p>The component uses a single 4KB block.</p>	
[3:0]	DES_2	<p>Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100.</p> <p><b>0b0100</b></p> <p>Arm Limited. This is bits[3:0] of the JEP106 continuation code.</p>	

### B.4.13 EDPIDR0, External Debug Peripheral Identification Register 0

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

## Configurations

- This register is available in all configurations.

## Attributes

## Width

32

## Functional group

## Debug

## Register offset

0xFE0

## Reset value

See individual bit resets.

## Bit descriptions

**Figure B-80: ext\_EDPIDR0 bit assignments**



### Table B-84: EDPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PART_0	Part number, least significant byte. <b>0b01001000</b> Least Significant byte of the debug part number	

## B.4.14 EDPIDR1, External Debug Peripheral Identification Register 1

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

Debug

#### Register offset

0xFE4

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-81: ext\_EDPIDR1 bit assignments**



**Table B-85: EDPIDR1 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. For Arm Limited, this field is 0b1011. <b>0b1011</b> Arm Limited. This is the least significant nibble of JEP106 ID code.	
[3:0]	PART_1	Part number, most significant nibble. <b>0b1101</b> Part number, most significant nibble.	

## B.4.15 EDPIDR2, External Debug Peripheral Identification Register 2

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

## Configurations

- This register is available in all configurations.

## Attributes

### Width

32

### Functional group

Debug

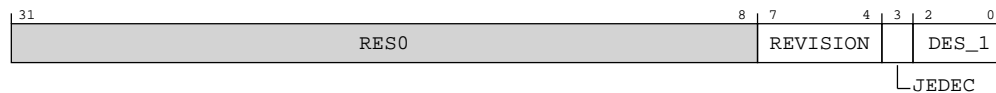
### Register offset

0xFE8

### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-82: ext\_EDPIDR2 bit assignments****Table B-86: EDPIDR2 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits. <b>0b0010</b> r2p1	
[3]	JEDEC	RAO. Indicates a JEP106 identity code is used. <b>0b1</b> RAO. Indicates a JEP106 identity code is used	
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code. For Arm Limited, this field is 0b011. <b>0b011</b> Arm Limited. This is bits[6:4] of the JEP106 ID code.	

## B.4.16 EDPIDR3, External Debug Peripheral Identification Register 3

Provides information to identify an external debug component.

For more information, see 'About the Peripheral identification scheme'.

## Configurations

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

Debug

**Register offset**

0xFEC

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-83: ext\_EDPIDR3 bit assignments****Table B-87: EDPIDR3 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	REVAND	Part minor revision. Parts using ext-EDPIDR2.REVISION as an extension to the Part number must use this field as a major revision number. <b>0b0000</b>	
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. <b>0b0000</b> The component is not modified from the original design.	

**B.4.17 EDCIDR0, External Debug Component Identification Register 0**

Provides information to identify an external debug component.

For more information, see 'About the Component Identification scheme'.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

Debug

Register offset

0xFF0

Reset value

See individual bit resets.

Bit descriptions

Figure B-84: ext\_EDCIDR0 bit assignments

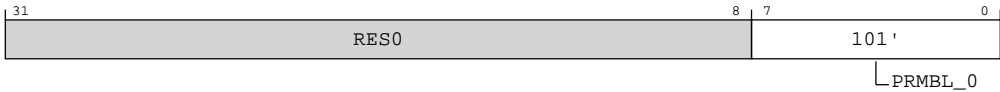


Table B-88: EDCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0

B.4.18 EDCIDR1, External Debug Component Identification Register 1

Provides information to identify an external debug component.

For more information, see 'About the Component Identification scheme'.

Configurations

- This register is available in all configurations.

Attributes

Width

32

Functional group

Debug

Register offset

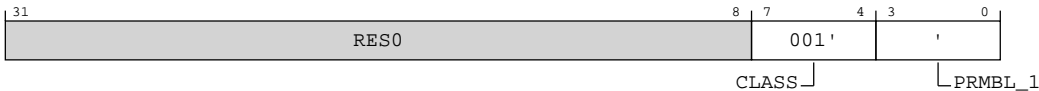
0xFF4

Reset value

See individual bit resets.

Bit descriptions

Figure B-85: ext\_EDCIDR1 bit assignments



### Table B-89: EDCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0

### B.4.19 EDCIDR2, External Debug Component Identification Register 2

Provides information to identify an external debug component.

For more information, see 'About the Component Identification scheme'.

## Configurations

- This register is available in all configurations.

## Attributes

## Width

32

## Functional group

## Debug

## Register offset

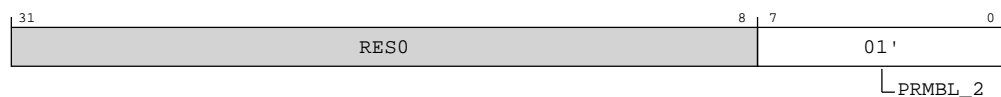
0xFF8

## Reset value

See individual bit resets.

## Bit descriptions

### Figure B-86: ext\_EDC IDR2 bit assignments



### Table B-90: EDCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0

#### B.4.20 EDCIDR3, External Debug Component Identification Register 3

Provides information to identify an external debug component.

For more information, see 'About the Component Identification scheme'.

## Configurations

- This register is available in all configurations.

## Attributes

### Width

32

### Functional group

Debug

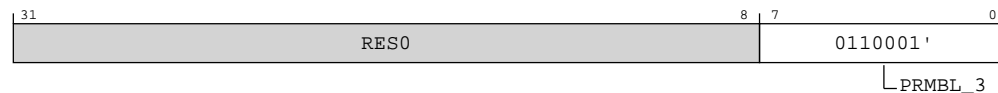
### Register offset

0xFFC

### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-87: ext\_EDCIDR3 bit assignments****Table B-91: EDCIDR3 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0

## B.5 AMU register summary

The summary table provides an overview of all AMU registers in the core. Individual register descriptions provide detailed information.

**Table B-92: AMU register summary**

Offset	Name	Reset	Width	Description
0x400	<a href="#">AMEVTYPERR00</a>	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x404	<a href="#">AMEVTYPERR01</a>	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x408	<a href="#">AMEVTYPERR02</a>	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x40C	<a href="#">AMEVTYPERR03</a>	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 0
0x480	<a href="#">AMEVTYPERR10</a>	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x484	<a href="#">AMEVTYPERR11</a>	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x488	<a href="#">AMEVTYPERR12</a>	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0x48C	<a href="#">AMEVTYPERR13</a>	See individual bit resets.	32-bit	Activity Monitors Event Type Registers 1
0xCE0	<a href="#">AMCGCR</a>	See individual bit resets.	32-bit	Activity Monitors Counter Group Configuration Register

Offset	Name	Reset	Width	Description
0xE00	AMCFGR	See individual bit resets.	32-bit	Activity Monitors Configuration Register
0xE08	AMIIDR	See individual bit resets.	32-bit	Activity Monitors Implementation Identification Register
0xFBC	AMDEVARCH	See individual bit resets.	32-bit	Activity Monitors Device Architecture Register
0xFCC	AMDEVTYPE	See individual bit resets.	32-bit	Activity Monitors Device Type Register
0xFD0	AMPIDR4	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 4
0xFE0	AMPIDR0	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 0
0xFE4	AMPIDR1	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 1
0xFE8	AMPIDR2	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 2
0xFEC	AMPIDR3	See individual bit resets.	32-bit	Activity Monitors Peripheral Identification Register 3
0xFF0	AMCIDR0	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 0
0xFF4	AMCIDR1	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 1
0xFF8	AMCIDR2	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 2
0xFFC	AMCIDR3	See individual bit resets.	32-bit	Activity Monitors Component Identification Register 3

### B.5.1 AMEVTYPER00, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR00\_ELO counts.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

AMU

##### Register offset

0x400

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-88: ext\_AMEVTYPER03 bit assignments**





**Table B-93: AMEVTYPER00 bit descriptions**

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally required for each architected counter.  The following table shows the mapping between required event numbers and the corresponding counters:	

## B.5.2 AMEVTYPER01, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR01\_ELO counts.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

AMU

#### Register offset

0x404

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-89: ext\_AMEVTYPER03 bit assignments****Table B-94: AMEVTYPER01 bit descriptions**

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally required for each architected counter.  The following table shows the mapping between required event numbers and the corresponding counters:	

### B.5.3 AMEVTYPER02, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR02\_ELO counts.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

AMU

##### Register offset

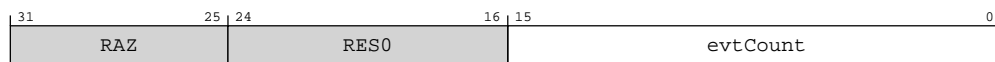
0x408

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-90: ext\_AMEVTYPER03 bit assignments**



**Table B-95: AMEVTYPER02 bit descriptions**

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally required for each architected counter.  The following table shows the mapping between required event numbers and the corresponding counters:	

### B.5.4 AMEVTYPER03, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR03\_ELO counts.

#### Configurations

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

AMU

**Register offset**

0x40C

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-91: ext\_AMEVTYPER03 bit assignments****Table B-96: AMEVTYPER03 bit descriptions**

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the architected activity monitor event counter ext-AMEVCNTR0<n>. The value of this field is architecturally required for each architected counter.  The following table shows the mapping between required event numbers and the corresponding counters:	

## B.5.5 AMEVTYPER10, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR10\_ELO counts.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

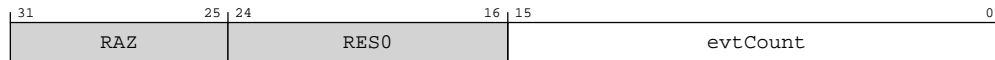
AMU

**Register offset**

0x480

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-92: ext\_AMEVTYPER13 bit assignments****Table B-97: AMEVTYPER10 bit descriptions**

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b0
[15:0]	evtCount	<p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1&lt;n&gt;.</p> <p>It is <b>IMPLEMENTATION DEFINED</b> what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter ext-AMEVCNTR1&lt;n&gt;, then:</p> <ul style="list-style-type: none"> <li>It is UNPREDICTABLE which event will be counted.</li> <li>The value read back is UNKNOWN.</li> </ul> <p><b>Note:</b> The event counted by ext-AMEVCNTR1&lt;n&gt; might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED.</p> <p>If the corresponding counter ext-AMEVCNTR1&lt;n&gt; is enabled, writes to this register have UNPREDICTABLE results.</p>	

## B.5.6 AMEVTYPER11, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR11\_ELO counts.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

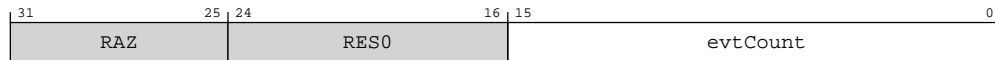
AMU

**Register offset**

0x484

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-93: ext\_AMEVTYPER13 bit assignments****Table B-98: AMEVTYPER11 bit descriptions**

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b0
[15:0]	evtCount	<p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1&lt;n&gt;.</p> <p>It is <b>IMPLEMENTATION DEFINED</b> what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter ext-AMEVCNTR1&lt;n&gt;, then:</p> <ul style="list-style-type: none"> <li>It is UNPREDICTABLE which event will be counted.</li> <li>The value read back is UNKNOWN.</li> </ul> <p><b>Note:</b> The event counted by ext-AMEVCNTR1&lt;n&gt; might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED.</p> <p>If the corresponding counter ext-AMEVCNTR1&lt;n&gt; is enabled, writes to this register have UNPREDICTABLE results.</p>	

## B.5.7 AMEVTYPER12, Activity Monitors Event Type Registers 1

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR12\_ELO counts.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

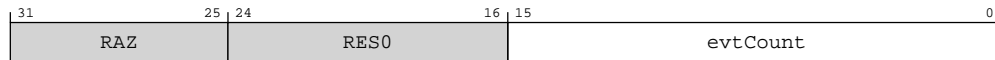
AMU

**Register offset**

0x488

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-94: ext\_AMEVTYPER13 bit assignments****Table B-99: AMEVTYPER12 bit descriptions**

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b0
[15:0]	evtCount	<p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1&lt;n&gt;.</p> <p>It is <b>IMPLEMENTATION DEFINED</b> what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter ext-AMEVCNTR1&lt;n&gt;, then:</p> <ul style="list-style-type: none"> <li>It is UNPREDICTABLE which event will be counted.</li> <li>The value read back is UNKNOWN.</li> </ul> <p><b>Note:</b> The event counted by ext-AMEVCNTR1&lt;n&gt; might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED.</p> <p>If the corresponding counter ext-AMEVCNTR1&lt;n&gt; is enabled, writes to this register have UNPREDICTABLE results.</p>	

**B.5.8 AMEVTYPER13, Activity Monitors Event Type Registers 1**

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR13\_ELO counts.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

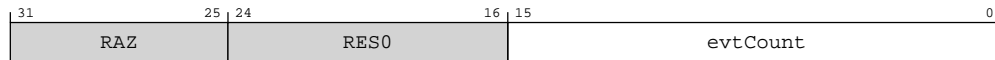
AMU

**Register offset**

0x48C

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-95: ext\_AMEVTYPER13 bit assignments****Table B-100: AMEVTYPER13 bit descriptions**

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b0
[15:0]	evtCount	<p>Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter ext-AMEVCNTR1&lt;n&gt;.</p> <p>It is <b>IMPLEMENTATION DEFINED</b> what values are supported by each counter.</p> <p>If software writes a value to this field which is not supported by the corresponding counter ext-AMEVCNTR1&lt;n&gt;, then:</p> <ul style="list-style-type: none"> <li>It is UNPREDICTABLE which event will be counted.</li> <li>The value read back is UNKNOWN.</li> </ul> <p><b>Note:</b> The event counted by ext-AMEVCNTR1&lt;n&gt; might be fixed at implementation. In this case, the field is read-only and writes are UNDEFINED.</p> <p>If the corresponding counter ext-AMEVCNTR1&lt;n&gt; is enabled, writes to this register have UNPREDICTABLE results.</p>	

## B.5.9 AMCGCR, Activity Monitors Counter Group Configuration Register

Provides information on the number of activity monitor event counters implemented within each counter group.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

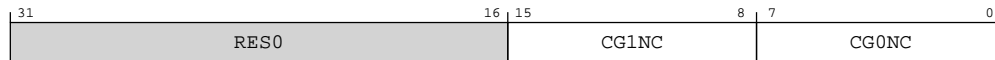
AMU

**Register offset**

0xCE0

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-96: ext\_AMCGCR bit assignments****Table B-101: AMCGCR bit descriptions**

Bits	Name	Description	Reset
[31:16]	RES0	Reserved	0b0
[15:8]	CG1NC	Counter Group 1 Number of Counters. The number of counters in the auxiliary counter group.  In AMUv1, the permitted range of values is 0 to 16.  <b>0b00000011</b> Three counters in the auxiliary counter group	
[7:0]	CG0NC	Counter Group 0 Number of Counters. The number of counters in the architected counter group.  In AMUv1, the value of this field is 4.  <b>0b00000100</b> Four Counters in the architected counter group	

## B.5.10 AMCFGR, Activity Monitors Configuration Register

Global configuration register for the activity monitors.

Provides information on supported features, the number of counter groups implemented, the total number of activity monitor event counters implemented, and the size of the counters. AMCFGR is applicable to both the architected and the auxiliary counter groups.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

AMU

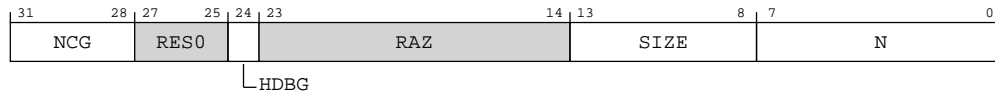
**Register offset**

0xE00



**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-97: ext\_AMCFGR bit assignments****Table B-102: AMCFGR bit descriptions**

Bits	Name	Description	Reset
[31:28]	NCG	Defines the number of counter groups. The following value is specified for this product. <b>0b0001</b> Two counter groups are implemented	
[27:25]	RES0	Reserved	0b0
[24]	HDBG	Halt-on-debug supported.  From Armv8, this feature must be supported, and so this bit is 0b1. <b>0b1</b> ext-AMCR.HDBG is read/write.	
[23:14]	RAZ	Reserved	
[13:8]	SIZE	Defines the size of activity monitor event counters.  The size of the activity monitor event counters implemented by the Activity Monitors Extension is defined as [AMCFGR.SIZE + 1].  From Armv8, the counters are 64-bit, and so this field is 0b111111.  <b>Note:</b> Software also uses this field to determine the spacing of counters in the memory-map. From Armv8, the counters are at doubleword-aligned addresses. <b>0b111111</b> 64 bits.	
[7:0]	N	Defines the number of activity monitor event counters.  The total number of counters implemented in all groups by the Activity Monitors Extension is defined as [AMCFGR.N + 1]. <b>0b00000110</b> Seven activity monitor event counters	

## B.5.11 AMIIDR, Activity Monitors Implementation Identification Register

Defines the implementer and revisions of the AMU.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

AMU

#### Register offset

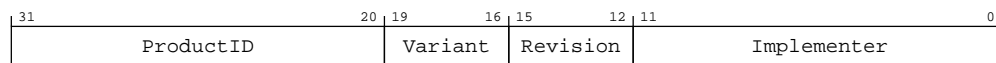
0xE08

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-98: ext\_AMIIDR bit assignments**



**Table B-103: AMIIDR bit descriptions**

Bits	Name	Description	Reset
[31:20]	ProductID	This field is an AMU part identifier. <b>0b110101001000</b> Cortex-X2	
[19:16]	Variant	This field distinguishes product variants or major revisions of the product. <b>0b0010</b> r2p1	
[15:12]	Revision	This field distinguishes minor revisions of the product. <b>0b0001</b> r2p1	
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the AMU.  For an Arm implementation, this field reads as 0x43B.	

## B.5.12 AMDEVARCH, Activity Monitors Device Architecture Register

Identifies the programmers' model architecture of the AMU component.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

AMU

#### Register offset

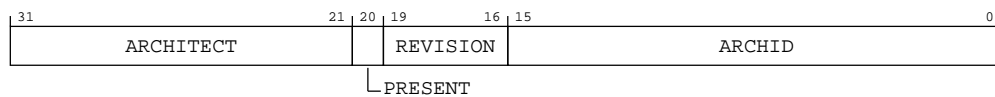
0xFBC

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-99: ext\_AMDEVARCH bit assignments**



**Table B-104: AMDEVARCH bit descriptions**

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For AMU, this is Arm Limited.	
[20]	PRESENT	When set to 1, indicates that the DEVARCH is present. <b>0b1</b> DEVARCH is present	
[19:16]	REVISION	Defines the architecture revision. For architectures defined by Arm this is the minor revision. <b>0b0000</b> Architecture revision is AMUv1.	
[15:0]	ARCHID	Defines this part to be an AMU component. For architectures defined by Arm this is further subdivided. For AMU: <ul style="list-style-type: none"> <li>Bits [15:12] are the architecture version, 0x0.</li> <li>Bits [11:0] are the architecture part number, 0xA66. This corresponds to AMU architecture version AMUv1.</li> </ul>	

B.5.13 AMDEVTTYPE, Activity Monitors Device Type Register

Indicates to a debugger that this component is part of a PE's performance monitor interface.

Configurations

- This register is available in all configurations.

Attributes

Width

32

Functional group

AMU

Register offset

0xFCC

Reset value

See individual bit resets.

Bit descriptions

Figure B-100: ext\_AMDEVTTYPE bit assignments



Table B-105: AMDEVTTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	SUB	Subtype. Reads as 0x1, to indicate this is a component within a PE.	
[3:0]	MAJOR	Major type. Reads as 0x6, to indicate this is a performance monitor component.	

B.5.14 AMPIDR4, Activity Monitors Peripheral Identification Register 4

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

Configurations

- This register is available in all configurations.

Attributes

Width

32

**Functional group**

AMU

**Register offset**

0xFD0

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-101: ext\_AMPIDR4 bit assignments****Table B-106: AMPIDR4 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	SIZE	4KB count.  <b>0b0000</b> The component uses a single 4KB block.	
[3:0]	DES_2	Designer. JEP106 continuation code, least significant nibble.  The value of this field is <b>IMPLEMENTATION DEFINED</b> . For Arm Limited, this field is 0b0100.  <b>0b0100</b> Arm Limited. This is bits[3:0] of the JEP106 continuation code.	

**B.5.15 AMPIDR0, Activity Monitors Peripheral Identification Register 0**

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

AMU

**Register offset**

0xFE0

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-102: ext\_AMPIDR0 bit assignments****Table B-107: AMPIDR0 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PART_0	Part number, least significant byte.  The value of this field is <b>IMPLEMENTATION DEFINED</b> .  <b>0b01001000</b> Part number, least significant byte.	

**B.5.16 AMPIDR1, Activity Monitors Peripheral Identification Register 1**

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

AMU

**Register offset**

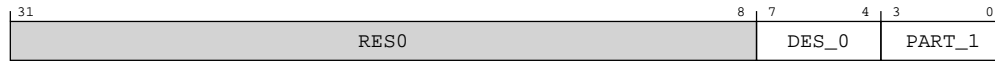
0xFE4

**Reset value**

See individual bit resets.

## Bit descriptions

**Figure B-103: ext\_AMPIDR1 bit assignments**



**Table B-108: AMPIDR1 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	DES_0	<p>Designer, least significant nibble of JEP106 ID code.</p> <p>The value of this field is <b>IMPLEMENTATION DEFINED</b>. For Arm Limited, this field is 0b1011.</p> <p><b>0b1011</b></p> <p>Designer, least significant nibble of JEP106 ID code.</p>	
[3:0]	PART_1	<p>Part number, most significant nibble.</p> <p>The value of this field is <b>IMPLEMENTATION DEFINED</b>.</p> <p><b>0b1101</b></p> <p>Part number, most significant nibble.</p>	

## B.5.17 AMPIDR2, Activity Monitors Peripheral Identification Register 2

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

AMU

#### Register offset

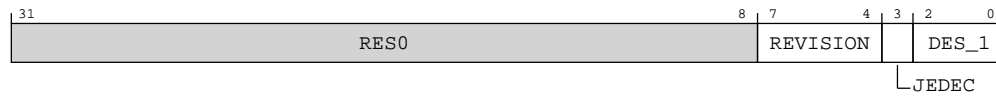
0xFE8

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-104: ext\_AMPIDR2 bit assignments**



**Table B-109: AMPIDR2 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	REVISION	Part major revision. Parts can also use this field to extend Part number to 16-bits.  The value of this field is <b>IMPLEMENTATION DEFINED</b> . <b>0b0010</b> r2p1	
[3]	JEDEC	RAO. Indicates a JEP106 identity code is used. <b>0b1</b> RAO. Indicates a JEP106 identity code is used.	
[2:0]	DES_1	Designer, most significant bits of JEP106 ID code.  The value of this field is <b>IMPLEMENTATION DEFINED</b> . For Arm Limited, this field is 0b011. <b>0b011</b> Arm Limited. This is bits[6:4] of the JEP106 ID code.	

## B.5.18 AMPIDR3, Activity Monitors Peripheral Identification Register 3

Provides information to identify an activity monitors component.

For more information, see 'About the Peripheral identification scheme'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

AMU

#### Register offset

0xFEC

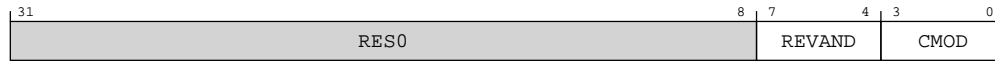
#### Reset value

See individual bit resets.



## Bit descriptions

**Figure B-105: ext\_AMPIDR3 bit assignments**



**Table B-110: AMPIDR3 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	REVAND	Part minor revision. Parts using ext-AMPIDR2.REVISION as an extension to the Part number must use this field as a major revision number.  The value of this field is <b>IMPLEMENTATION DEFINED</b> . <b>0b0000</b>	
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component.  The value of this field is <b>IMPLEMENTATION DEFINED</b> . <b>0b0000</b> The component is not modified from the original design.	

## B.5.19 AMCIDR0, Activity Monitors Component Identification Register 0

Provides information to identify an activity monitors component.

For more information, see 'About the Component identification scheme'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

AMU

#### Register offset

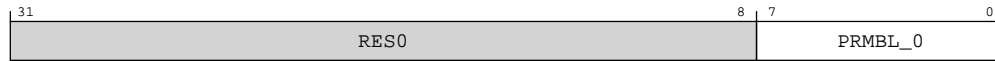
0xFF0

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-106: ext\_AMCIDR0 bit assignments**



**Table B-111: AMCIDR0 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_0	Preamble. Must read as 0x0D. <b>0b00001101</b> Preamble	

## B.5.20 AMCIDR1, Activity Monitors Component Identification Register 1

Provides information to identify an activity monitors component.

For more information, see 'About the Component identification scheme'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

AMU

#### Register offset

0xFF4

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-107: ext\_AMCIDR1 bit assignments**



**Table B-112: AMCIDR1 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	CLASS	Component class. Reads as 0x9, CoreSight component. <b>0b1001</b> CoreSight component.	
[3:0]	PRMBL_1	Preamble. Reads as 0x0. <b>0b0000</b> Preamble	

## B.5.21 AMCIDR2, Activity Monitors Component Identification Register 2

Provides information to identify an activity monitors component.

For more information, see 'About the Component identification scheme'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

AMU

#### Register offset

0xFF8

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-108: ext\_AMCIDR2 bit assignments****Table B-113: AMCIDR2 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_2	Preamble. Reads as 0x05. <b>0b00000101</b> Preamble byte 2	

## B.5.22 AMCIDR3, Activity Monitors Component Identification Register 3

Provides information to identify an activity monitors component.

For more information, see 'About the Component identification scheme'.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

AMU

#### Register offset

0xFFC

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-109: ext\_AMCIDR3 bit assignments**



**Table B-114: AMCIDR3 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_3	Preamble. Reads as 0xB1. <b>0b10110001</b> Preamble byte 3	

## B.6 ETE register summary

The summary table provides an overview of all ETE registers in the core. Individual register descriptions provide detailed information.

**Table B-115: ETE register summary**

Offset	Name	Reset	Width	Description
0x018	TRCAUXCTLR	0x0	32-bit	Auxillary Control Register

Offset	Name	Reset	Width	Description
0x180	TRCIDR8	See individual bit resets.	32-bit	ID Register 8
0x184	TRCIDR9	See individual bit resets.	32-bit	ID Register 9
0x188	TRCIDR10	See individual bit resets.	32-bit	ID Register 10
0x18C	TRCIDR11	See individual bit resets.	32-bit	ID Register 11
0x190	TRCIDR12	0x0	32-bit	ID Register 12
0x194	TRCIDR13	0x0	32-bit	ID Register 13
0x1C0	TRCIMSPEC0	See individual bit resets.	32-bit	IMP DEF Register 0
0x1E0	TRCIDR0	See individual bit resets.	32-bit	ID Register 0
0x1E4	TRCIDR1	See individual bit resets.	32-bit	ID Register 1
0x1E8	TRCIDR2	See individual bit resets.	32-bit	ID Register 2
0x1EC	TRCIDR3	See individual bit resets.	32-bit	ID Register 3
0x1F0	TRCIDR4	See individual bit resets.	32-bit	ID Register 4
0x1F4	TRCIDR5	See individual bit resets.	32-bit	ID Register 5
0x1F8	TRCIDR6	0x0	32-bit	ID Register 6
0x1FC	TRCIDR7	0x0	32-bit	ID Register 7
0xF00	TRCITCTRL	See individual bit resets.	32-bit	Integration Mode Control Register
0xFA0	TRCCLAIMSET	See individual bit resets.	32-bit	Claim Tag Set Register
0xFA4	TRCCLAIMCLR	See individual bit resets.	32-bit	Claim Tag Clear Register
0xFBC	TRCDEVARCH	See individual bit resets.	32-bit	Device Architecture Register
0xFC0	TRCDEVID2	0x0	32-bit	Device Configuration Register 2
0xFC4	TRCDEVID1	0x0	32-bit	Device Configuration Register 1
0xFC8	TRCDEVID	0x0	32-bit	Device Configuration Register
0xFCC	TRCDEVTYPE	See individual bit resets.	32-bit	Device Type Register
0xFD0	TRCPIDR4	See individual bit resets.	32-bit	Peripheral Identification Register 4
0xFD4	TRCPIDR5	0x0	32-bit	Peripheral Identification Register 5
0xFD8	TRCPIDR6	0x0	32-bit	Peripheral Identification Register 6
0xFDC	TRCPIDR7	0x0	32-bit	Peripheral Identification Register 7
0xFE0	TRCPIDR0	See individual bit resets.	32-bit	Peripheral Identification Register 0
0xFE4	TRCPIDR1	See individual bit resets.	32-bit	Peripheral Identification Register 1
0xFE8	TRCPIDR2	See individual bit resets.	32-bit	Peripheral Identification Register 2
0xFEC	TRCPIDR3	See individual bit resets.	32-bit	Peripheral Identification Register 3
0xFF0	TRCCIDR0	See individual bit resets.	32-bit	Component Identification Register 0
0xFF4	TRCCIDR1	See individual bit resets.	32-bit	Component Identification Register 1
0xFF8	TRCCIDR2	See individual bit resets.	32-bit	Component Identification Register 2
0xFFC	TRCCIDR3	See individual bit resets.	32-bit	Component Identification Register 3

### B.6.1 TRCAUXCTLR, Auxillary Control Register

The function of this register is **IMPLEMENTATION DEFINED**.

#### Configurations

- This register is available in all configurations.

#### Attributes

**Width**

32

**Functional group**

ETE

**Register offset**

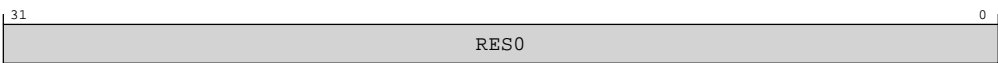
0x018

**Reset value**

0x0

#### Bit descriptions

**Figure B-110: ext\_TRCAUXCTLR bit assignments**



**Table B-116: TRCAUXCTLR bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

### B.6.2 TRCIDR8, ID Register 8

Returns the maximum speculation depth of the instruction trace element stream.

#### Configurations

- This register is available in all configurations.

#### Attributes

**Width**

32

**Functional group**

ETE

Register offset

0x180

Reset value

See individual bit resets.

Bit descriptions

Figure B-111: ext\_TRCIDR8 bit assignments



Table B-117: TRCIDR8 bit descriptions

Bits	Name	Description	Reset
[31:0]	MAXSPEC	Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of PO elements in the trace element stream that can be speculative at any time.	

B.6.3 TRCIDR9, ID Register 9

Returns the tracing capabilities of the trace unit.

Configurations

- This register is available in all configurations.

Attributes

Width

32

Functional group

ETE

Register offset

0x184

Reset value

See individual bit resets.

Bit descriptions

Figure B-112: ext\_TRCIDR9 bit assignments



### Table B-118: TRCIDR9 bit descriptions

Bits	Name	Description	Reset
[31:0]	NUMPOKEY	Indicates the number of P0 right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.	

### B.6.4 TRCIDR10, ID Register 10

Returns the tracing capabilities of the trace unit.

## Configurations

- This register is available in all configurations.

## Attributes

## Width

32

### Functional group

FTF

## Register offset

0x188

## Reset value

See individual bit resets.

## Bit descriptions

**Figure B-113: ext\_TRCIDR10 bit assignments**



### Table B-119: TRCIDR10 bit descriptions

Bits	Name	Description	Reset
[31:0]	NUMP1KEY	Indicates the number of P1 right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.	

### B.6.5 TRCIDR11, ID Register 11

Returns the tracing capabilities of the trace unit.

## Configurations

- This register is available in all configurations.



**Attributes****Width**

32

**Functional group**

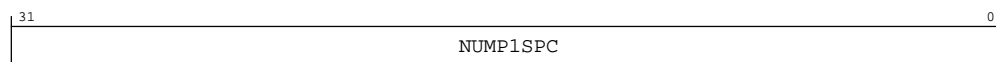
ETE

**Register offset**

0x18C

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-114: ext\_TRCIDR11 bit assignments****Table B-120: TRCIDR11 bit descriptions**

Bits	Name	Description	Reset
[31:0]	NUMP1SPC	Indicates the number of special P1 right-hand keys. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures.	

**B.6.6 TRCIDR12, ID Register 12**

Returns the tracing capabilities of the trace unit.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

ETE

**Register offset**

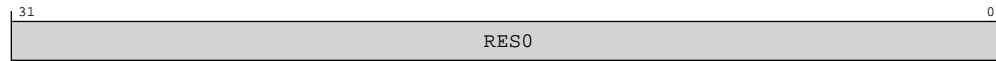
0x190

**Reset value**

0x0

## Bit descriptions

**Figure B-115: ext\_TRCIDR12 bit assignments**



**Table B-121: TRCIDR12 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

## B.6.7 TRCIDR13, ID Register 13

Returns the tracing capabilities of the trace unit.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

ETE

#### Register offset

0x194

#### Reset value

0x0

## Bit descriptions

**Figure B-116: ext\_TRCIDR13 bit assignments**



**Table B-122: TRCIDR13 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

## B.6.8 TRCIMSPEC0, IMP DEF Register 0

TRCIMSPEC0 shows the presence of any **IMPLEMENTATION DEFINED** features, and provides an interface to enable the features that are provided.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

ETE

#### Register offset

0x1C0

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-117: ext\_TRCIMSPEC0 bit assignments**



**Table B-123: TRCIMSPEC0 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	EN	Enable. Controls whether the <b>IMPLEMENTATION DEFINED</b> features are enabled.  <b>0b0000</b> The <b>IMPLEMENTATION DEFINED</b> features are not enabled. The trace unit must behave as if the <b>IMPLEMENTATION DEFINED</b> features are not supported.	
[3:0]	SUPPORT	Indicates whether the implementation supports <b>IMPLEMENTATION DEFINED</b> features.  <b>0b0000</b> No <b>IMPLEMENTATION DEFINED</b> features are supported.	

## B.6.9 TRCIDR0, ID Register 0

Returns the tracing capabilities of the trace unit.

### Configurations

- This register is available in all configurations.

## Attributes

### Width

32

### Functional group

ETE

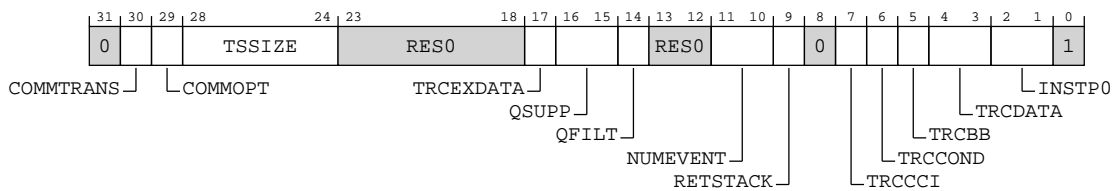
### Register offset

0x1E0

### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-118: ext\_TRCIDR0 bit assignments****Table B-124: TRCIDR0 bit descriptions**

Bits	Name	Description	Reset
[31]	RES0	Reserved	0b0
[30]	COMMTRANS	Transaction Start element behavior. <b>0b0</b> Transaction Start elements are P0 elements.	
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. <b>0b1</b> Commit mode 1.	
[28:24]	TSSIZE	Indicates that the trace unit implements Global timestamping and the size of the timestamp value. <b>0b01000</b> Global timestamping implemented with a 64-bit timestamp value.	
[23:18]	RES0	Reserved	0b0
[17]	TRCEXDATA	Indicates if the trace unit implements tracing of data transfers for exceptions and exception returns. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures. <b>0b0</b> Tracing of data transfers for exceptions and exception returns not implemented. <b>0b1</b> Tracing of data transfers for exceptions and exception returns implemented.	

Bits	Name	Description	Reset
[16:15]	QSUPP	Indicates that the trace unit implements Q element support.  <b>0b00</b> Q element support is not implemented.	
[14]	QFILT	Indicates if the trace unit implements Q element filtering.  <b>0b0</b> Q element filtering is not implemented.	
[13:12]	RES0	Reserved	0b0
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented.  <b>0b11</b> The trace unit supports 4 ETEEvents.	
[9]	RETSTACK	Indicates if the trace unit supports the return stack.  <b>0b1</b> Return stack implemented.	
[8]	RES0	Reserved	0b0
[7]	TRCCCI	Indicates if the trace unit implements cycle counting.  <b>0b1</b> Cycle counting implemented.	
[6]	TRCCOND	Indicates if the trace unit implements conditional instruction tracing. Conditional instruction tracing is not implemented in ETE and this field is reserved for other trace architectures.  <b>0b0</b> Conditional instruction tracing not implemented.	
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting.  <b>0b1</b> Branch broadcasting implemented.	
[4:3]	TRCDATA	Indicates if the trace unit implements data tracing. Data tracing is not implemented in ETE and this field is reserved for other trace architectures.  <b>0b00</b> Data tracing not implemented.	
[2:1]	INSTPO	Indicates if load and store instructions are PO instructions. Load and store instructions as PO instructions is not implemented in ETE and this field is reserved for other trace architectures.  <b>0b00</b> Load and store instructions are not PO instructions.	
[0]	RES1	Reserved	0b1

## B.6.10 TRCIDR1, ID Register 1

Returns the tracing capabilities of the trace unit.

### Configurations

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

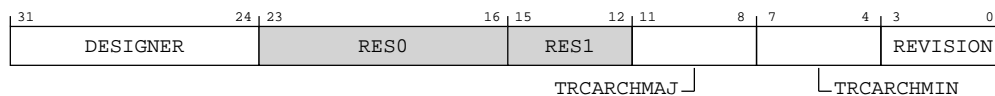
ETE

**Register offset**

0x1E4

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-119: ext\_TRCIDR1 bit assignments****Table B-125: TRCIDR1 bit descriptions**

Bits	Name	Description	Reset
[31:24]	DESIGNER	Indicates which company designed the trace unit. The permitted values of this field are the same as AArch64-MIDR_EL1.Implementer. <b>0b01000001</b> Arm Limited	
[23:16]	RES0	Reserved	0b0
[15:12]	RES1	Reserved	0b1
[11:8]	TRCARCHMAJ	Major architecture version. <b>0b1111</b> If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH.	
[7:4]	TRCARCHMIN	Minor architecture version. <b>0b1111</b> If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH.	
[3:0]	REVISION	Implementation revision that identifies the revision of the trace and OS Lock registers. <b>0b0010</b> Revision 2	

**B.6.11 TRCIDR2, ID Register 2**

Returns the tracing capabilities of the trace unit.

**Configurations**

- This register is available in all configurations.

## Attributes

### Width

32

### Functional group

ETE

### Register offset

0x1E8

### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-120: ext\_TRCIDR2 bit assignments****Table B-126: TRCIDR2 bit descriptions**

Bits	Name	Description	Reset
[31]	WFXMODE	Indicates whether WFI and WFE instructions are classified as P0 instructions. <b>0b1</b> WFI and WFE instructions are classified as P0 instructions.	
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. <b>0b10</b> Virtual context identifier selection not supported. ext-TRCCONFIGR.VMIDOPT is RES1.	
[28:25]	CCSIZE	Indicates the size of the cycle counter. <b>0b0000</b> The cycle counter is 12 bits in length.	
[24:20]	DVSIZE	Indicates the data value size in bytes. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures. <b>0b01000</b> Data value tracing has a maximum of 64-bit data values.	
[19:15]	DASIZE	Indicates the data value size in bytes. Data tracing is not implemented in ETE and this field is reserved for other trace architectures. Allocated in other trace architectures. <b>0b01000</b> Data address tracing has a maximum of 64-bit data addresses.	
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. <b>0b00100</b> 32-bit Virtual context identifier size.	
[9:5]	CIDSIZE	Indicates the Context identifier size. <b>0b00100</b> 32-bit Context identifier size.	

Bits	Name	Description	Reset
[4:0]	IASIZE	Virtual instruction address size.  <b>0b01000</b> Maximum of 64-bit instruction address size.	

## B.6.12 TRCIDR3, ID Register 3

Returns the base architecture of the trace unit.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

ETE

#### Register offset

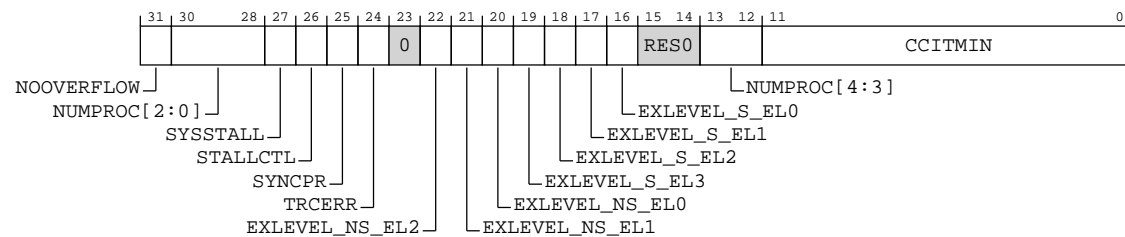
0x1EC

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-121: ext\_TRCIDR3 bit assignments**



**Table B-127: TRCIDR3 bit descriptions**

Bits	Name	Description	Reset
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented.  <b>0b0</b> Overflow prevention is not implemented.	
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing.  <b>0b00000</b> The trace unit can trace one PE.	



Bits	Name	Description	Reset
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. <b>0b0</b> Stalling of the PE is not permitted.	
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. <b>0b0</b> Stalling of the PE is not implemented.	
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. <b>0b0</b> ext-TRCSYNCPR is read-write so software can change the synchronization period.	
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. <b>0b1</b> Forced tracing of System Error exceptions is implemented.	
[23]	RES0	Reserved	0b0
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 implemented. <b>0b1</b> Non-secure EL2 is implemented.	
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 implemented. <b>0b1</b> Non-secure EL1 is implemented.	
[20]	EXLEVEL_NS_ELO	Indicates if Non-secure ELO implemented. <b>0b1</b> Non-secure ELO is implemented.	
[19]	EXLEVEL_S_EL3	Indicates if Secure EL3 implemented. <b>0b1</b> Secure EL3 is implemented.	
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 implemented. <b>0b1</b> Secure EL2 is implemented.	
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 implemented. <b>0b1</b> Secure EL1 is implemented.	
[16]	EXLEVEL_S_ELO	Indicates if Secure ELO implemented. <b>0b1</b> Secure ELO is implemented.	
[15:14]	RES0	Reserved	0b0
[11:0]	CCITMIN	Indicates the minimum value that can be programmed in ext-TRCCCCTLR.THRESHOLD.  If ext-TRCIDR0.TRCCCI == 0b1 then the minimum value of this field is 0x001.  If ext-TRCIDR0.TRCCCI == 0b0 then this field is zero.	

### B.6.13 TRCIDR4, ID Register 4

Returns the tracing capabilities of the trace unit.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

ETE

##### Register offset

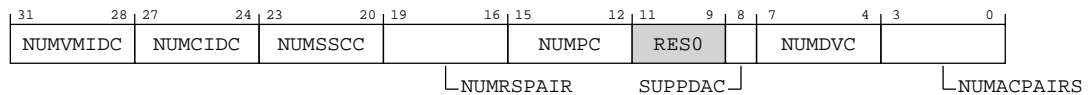
0x1F0

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-122: ext\_TRCIDR4 bit assignments**



**Table B-128: TRCIDR4 bit descriptions**

Bits	Name	Description	Reset
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. <b>0b0001</b> The implementation has one Virtual Context Identifier Comparator.	
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. <b>0b0001</b> The implementation has one Context Identifier Comparator.	
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. <b>0b0001</b> The implementation has one Single-shot Comparator Control.	
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. <b>0b0111</b> The implementation has eight resource selector pairs.	
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. <b>0b0000</b> No PE Comparator Inputs are available.	
[11:9]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[8]	SUPPDAC	Indicates whether data address comparisons are implemented. Data address comparisons are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures.  <b>0b0</b> Data address comparisons not implemented.	
[7:4]	NUMDVC	Indicates the number of data value comparators. Data value comparators are not implemented in ETE and are reserved for other trace architectures. Allocated in other trace architectures.  <b>0b0000</b> No data value comparators implemented.	
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing.  <b>0b0100</b> The implementation has four Address Comparator pairs.	

### B.6.14 TRCIDR5, ID Register 5

Returns the tracing capabilities of the trace unit.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

ETE

##### Register offset

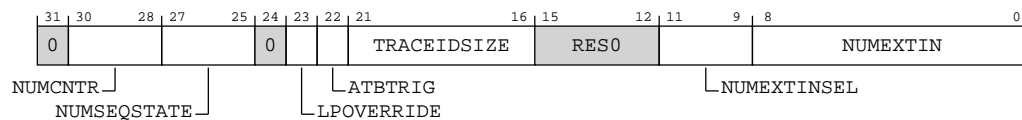
0x1F4

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-123: ext\_TRCIDR5 bit assignments**



**Table B-129: TRCIDR5 bit descriptions**

Bits	Name	Description	Reset
[31]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[30:28]	NUMCNTR	Indicates the number of Counters that are available for tracing. <b>0b010</b> Two Counters implemented.	
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. <b>0b100</b> Four Sequencer states are implemented.	
[24]	RES0	Reserved	0b0
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. <b>0b0</b> The trace unit does not support Low-power Override Mode.	
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. <b>0b1</b> The implementation supports ATB triggers.	
[21:16]	TRACEIDSIZE	Indicates the trace ID width. <b>0b000111</b> The implementation supports a 7-bit trace ID.	
[15:12]	RES0	Reserved	0b0
[11:9]	NUMEXTINSEL	Indicates how many External Input Selector resources are implemented. <b>0b100</b> 4 External Input Selector resources are available.	
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. <b>0b11111111</b> Unified PMU event selection.	

### B.6.15 TRCIDR6, ID Register 6

Returns the tracing capabilities of the trace unit.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

ETE

##### Register offset

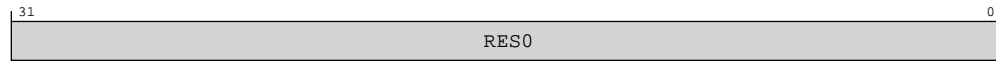
0x1F8

##### Reset value

0x0

## Bit descriptions

**Figure B-124: ext\_TRCIDR6 bit assignments**



**Table B-130: TRCIDR6 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

## B.6.16 TRCIDR7, ID Register 7

Returns the tracing capabilities of the trace unit.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

ETE

#### Register offset

0x1FC

#### Reset value

0x0

## Bit descriptions

**Figure B-125: ext\_TRCIDR7 bit assignments**



**Table B-131: TRCIDR7 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

## B.6.17 TRCITCTRL, Integration Mode Control Register

A component can use TRCITCTRL to dynamically switch between functional mode and integration mode. In integration mode, topology detection is enabled. After switching to integration mode and performing integration tests or topology detection, reset the system to ensure correct behavior of CoreSight and other connected system components.

For additional information see the CoreSight Architecture Specification.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

ETE

#### Register offset

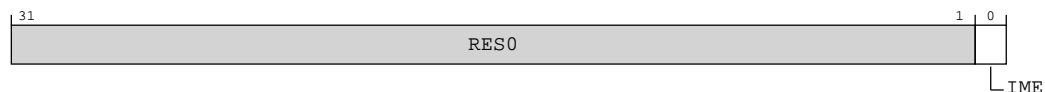
0xF00

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-126: ext\_TRCITCTRL bit assignments**



**Table B-132: TRCITCTRL bit descriptions**

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	0b0
[0]	IME	Integration Mode Enable.  <b>0b0</b> The component must enter functional mode.  <b>0b1</b> The component must enter integration mode, and enable support for topology detection and integration testing.	

## B.6.18 TRCCLAIMSET, Claim Tag Set Register

In conjunction with ext-TRCCLAIMCLR, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information see the CoreSight Architecture Specification.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

ETE

#### Register offset

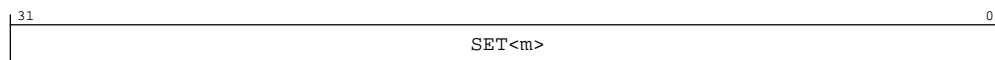
0xFA0

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-127: ext\_TRCCLAIMSET bit assignments**



**Table B-133: TRCCLAIMSET bit descriptions**

Bits	Name	Description	Reset
[31:0]	SET<m>	<p>Claim Tag Set. Indicates whether Claim Tag bit m is implemented, and is used to set Claim Tag bit m to 0b1.</p> <p><b>0b0</b></p> <p>On a read: Claim Tag bit m is not implemented.</p> <p>On a write: Ignored.</p> <p><b>0b1</b></p> <p>On a read: Claim Tag bit m is implemented.</p> <p>On a write: Set Claim Tag bit m to 0b1.</p>	

### B.6.19 TRCCLAIMCLR, Claim Tag Clear Register

In conjunction with ext-TRCCLAIMSET, provides Claim Tag bits that can be separately set and cleared to indicate whether functionality is in use by a debug agent.

For additional information see the CoreSight Architecture Specification.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

ETE

##### Register offset

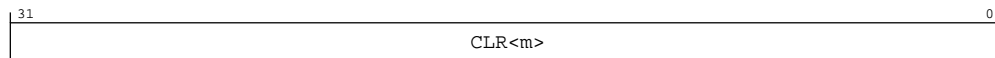
0xFA4

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-128: ext\_TRCCLAIMCLR bit assignments**



**Table B-134: TRCCLAIMCLR bit descriptions**

Bits	Name	Description	Reset
[31:0]	CLR<m>	<p>Claim Tag Clear. Indicates the current status of the Claim Tag bit m, and is used to clear Claim Tag bit m to 0b0.</p> <p><b>0b0</b></p> <p>On a read: Claim Tag bit m is not set.</p> <p>On a write: Ignored.</p> <p><b>0b1</b></p> <p>On a read: Claim Tag bit m is set.</p> <p>On a write: Clear Claim tag bit m to 0b0.</p>	

### B.6.20 TRCDEVARCH, Device Architecture Register

Provides discovery information for the component.

For additional information see the CoreSight Architecture Specification.



## Configurations

- This register is available in all configurations.

## Attributes

### Width

32

### Functional group

ETE

### Register offset

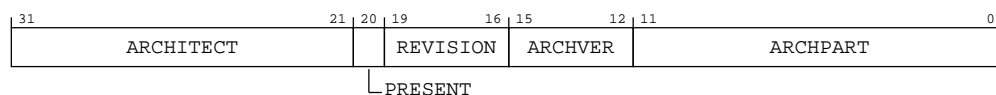
0xFBC

### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-129: ext\_TRCDEVARCH bit assignments**



**Table B-135: TRCDEVARCH bit descriptions**

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architect of the component. Bits [31:28] are the JEP106 continuation code (JEP106 bank ID, minus 1) and bits [27:21] are the JEP106 ID code. <b>0b01000111011</b> JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.	
[20]	PRESENT	DEVARCH Present. Defines that the DEVARCH register is present. <b>0b1</b> Device Architecture information present.	
[19:16]	REVISION	Revision. Defines the architecture revision of the component. <b>0b0000</b> ETE Version 1.0.	
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. <b>0b0101</b> ETE Version 1.	
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. <b>0b101000010011</b> Arm PE trace architecture.	

B.6.21 TRCDEVID2, Device Configuration Register 2

Provides discovery information for the component.

For additional information see the CoreSight Architecture Specification.

Configurations

- This register is available in all configurations.

Attributes

Width

32

Functional group

ETE

Register offset

0xFC0

Reset value

0x0

Bit descriptions

Figure B-130: ext\_TRCDEVID2 bit assignments

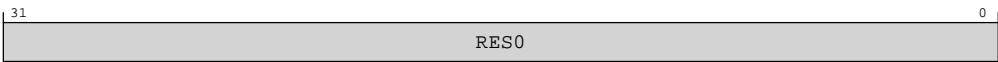


Table B-136: TRCDEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

B.6.22 TRCDEVID1, Device Configuration Register 1

Provides discovery information for the component.

For additional information see the CoreSight Architecture Specification.

Configurations

- This register is available in all configurations.

Attributes

Width

32

Functional group

ETE

Register offset

0xFC4

Reset value

0x0

Bit descriptions

Figure B-131: ext\_TRCDEVID1 bit assignments

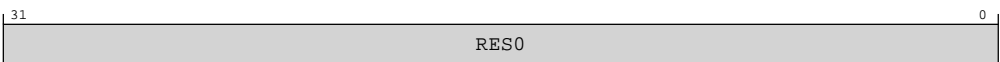


Table B-137: TRCDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

B.6.23 TRCDEVID, Device Configuration Register

Provides discovery information for the component.

For additional information see the CoreSight Architecture Specification.

Configurations

- This register is available in all configurations.

Attributes

Width

32

Functional group

ETE

Register offset

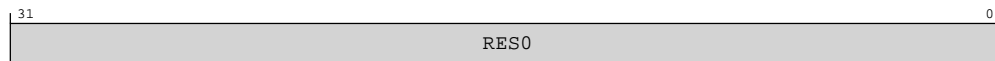
0xFC8

Reset value

0x0

## Bit descriptions

**Figure B-132: ext\_TRCDEVID bit assignments**



**Table B-138: TRCDEVID bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

## B.6.24 TRCDEVTYPE, Device Type Register

Provides discovery information for the component. If the part number field is not recognised, a debugger can report the information that is provided by TRCDEVTYPE about the component instead.

For additional information see the CoreSight Architecture Specification.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

ETE

#### Register offset

0xFCC

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-133: ext\_TRCDEVTYPE bit assignments**



**Table B-139: TRCDEVTYPE bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[7:4]	SUB	Component sub-type. <b>0b0001</b> When MAJOR == 0x3 (Trace source): Associated with a PE.	
[3:0]	MAJOR	Component major type. <b>0b0011</b> Trace source.	

## B.6.25 TRCPIDR4, Peripheral Identification Register 4

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

ETE

#### Register offset

0xFD0

#### Reset value

See individual bit resets.

### Bit descriptions

**Figure B-134: ext\_TRCPIDR4 bit assignments**



**Table B-140: TRCPIDR4 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	SIZE	The component uses a single 4KB block. <b>0b0000</b>	
[3:0]	DES_2	Arm Limited. This is bits[3:0] of the JEP106 continuation code. <b>0b0100</b>	

B.6.26 TRCPIDR5, Peripheral Identification Register 5

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

- This register is available in all configurations.

Attributes

**Width**  
32

**Functional group**  
ETE

**Register offset**  
0xFD4

**Reset value**  
0x0

Bit descriptions

Figure B-135: ext\_TRCPIDR5 bit assignments

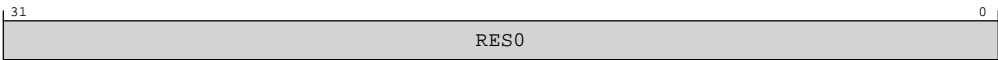


Table B-141: TRCPIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

B.6.27 TRCPIDR6, Peripheral Identification Register 6

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

- This register is available in all configurations.

Attributes

**Width**  
32

Functional group

ETE

Register offset

0xFD8

Reset value

0x0

Bit descriptions

Figure B-136: ext\_TRCPIDR6 bit assignments

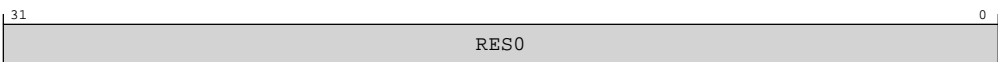


Table B-142: TRCPIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

B.6.28 TRCPIDR7, Peripheral Identification Register 7

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

- This register is available in all configurations.

Attributes

Width

32

Functional group

ETE

Register offset

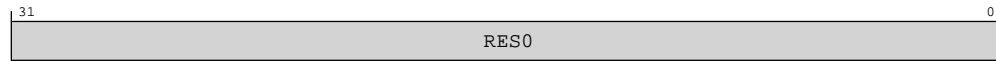
0xFDC

Reset value

0x0

## Bit descriptions

**Figure B-137: ext\_TRCPIDR7 bit assignments**



**Table B-143: TRCPIDR7 bit descriptions**

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

## B.6.29 TRCPIDR0, Peripheral Identification Register 0

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

### Configurations

- This register is available in all configurations.

### Attributes

#### Width

32

#### Functional group

ETE

#### Register offset

0xFE0

#### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-138: ext\_TRCPIDR0 bit assignments**



**Table B-144: TRCPIDR0 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PART_0	Least significant byte of the trace unit part. <b>0b01001000</b>	



### B.6.30 TRCPIDR1, Peripheral Identification Register 1

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

ETE

##### Register offset

0xFE4

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-139: ext\_TRCPIDR1 bit assignments**



**Table B-145: TRCPIDR1 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	DES_0	Arm Limited. This is the least significant nibble of JEP106 ID code. <b>0b1011</b>	
[3:0]	PART_1	Part number, most significant nibble. <b>0b1101</b>	

### B.6.31 TRCPIDR2, Peripheral Identification Register 2

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

## Configurations

- This register is available in all configurations.

## Attributes

### Width

32

### Functional group

ETE

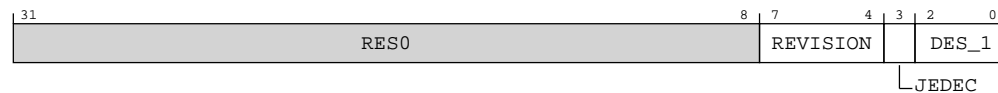
### Register offset

0xFE8

### Reset value

See individual bit resets.

## Bit descriptions

**Figure B-140: ext\_TRCPIDR2 bit assignments****Table B-146: TRCPIDR2 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	REVISION	r2p1 - Part major revision. <b>0b0010</b>	
[3]	JEDEC	JEDEC-assigned JEP106 implementer code is used. <b>0b1</b> RES1. Indicates a JEP106 identity code is used	
[2:0]	DES_1	Arm Limited. Most significant nibble of JEP106 ID code. <b>0b011</b>	

## B.6.32 TRCPIDR3, Peripheral Identification Register 3

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

## Configurations

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

ETE

**Register offset**

0xFEC

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-141: ext\_TRCPIDR3 bit assignments****Table B-147: TRCPIDR3 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	REVAND	Part minor revision. <b>0b0001</b>	
[3:0]	CMOD	Not Customer modified. <b>0b0000</b>	

**B.6.33 TRCCIDR0, Component Identification Register 0**

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

ETE

**Register offset**

0xFF0

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-142: ext\_TRCCIDR0 bit assignments****Table B-148: TRCCIDR0 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_0	Component identification preamble, segment 0.  <b>0b00001101</b> Preamble byte 0	

**B.6.34 TRCCIDR1, Component Identification Register 1**

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

**Configurations**

- This register is available in all configurations.

**Attributes****Width**

32

**Functional group**

ETE

**Register offset**

0xFF4

**Reset value**

See individual bit resets.

**Bit descriptions****Figure B-143: ext\_TRCCIDR1 bit assignments**

**Table B-149: TRCCIDR1 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	CLASS	Component class.  <b>0b1001</b> CoreSight peripheral.	
[3:0]	PRMBL_1	Component identification preamble, segment 1.  <b>0b0000</b> Preamble	

### B.6.35 TRCCIDR2, Component Identification Register 2

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

#### Configurations

- This register is available in all configurations.

#### Attributes

##### Width

32

##### Functional group

ETE

##### Register offset

0xFF8

##### Reset value

See individual bit resets.

#### Bit descriptions

**Figure B-144: ext\_TRCCIDR2 bit assignments****Table B-150: TRCCIDR2 bit descriptions**

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_2	Component identification preamble, segment 2.  <b>0b00000101</b> Preamble byte 2.	

B.6.36 TRCCIDR3, Component Identification Register 3

Provides discovery information about the component.

For additional information see the CoreSight Architecture Specification.

Configurations

- This register is available in all configurations.

Attributes

Width

32

Functional group

ETE

Register offset

0xFFC

Reset value

See individual bit resets.

Bit descriptions

Figure B-145: ext\_TRCCIDR3 bit assignments



Table B-151: TRCCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_3	Component identification preamble, segment 3. <b>0b10110001</b> Preamble byte 3.	

# Appendix C Document revisions

This appendix records the changes between released issues of this document.

## C.1 Revisions

Changes between released issues of this book are summarized in tables.

The first table is for the first release. Then, each table compares the new issue of the book with the last released issue of the book. Release numbers match the revision history in [Release Information](#) on page 2.

**Table C-1: Issue 0000-01**

Change	Location
First alpha release for r0p0	-

**Table C-2: Differences between Issue 0000-01 and Issue 0000-02**

Change	Location
First beta release for r0p0	Revisions history
Added ELA information	<ul style="list-style-type: none"> <li>• <a href="#">2.1 Cortex-X2 core features</a> on page 22</li> <li>• <a href="#">2.2 Cortex-X2 core configuration options</a> on page 24</li> <li>• <a href="#">3.1 Core components</a> on page 31</li> </ul>
Documented registers	<ul style="list-style-type: none"> <li>• <a href="#">13 Advanced SIMD and floating-point support</a> on page 93</li> <li>• <a href="#">15 System control</a> on page 95</li> </ul>
Added ROM table information	<a href="#">16 Debug</a> on page 97
Restructured and updated memory management information	<a href="#">6 Memory management</a> on page 49

**Table C-3: Differences between Issue 0000-02 and Issue 0000-03**

Change	Location
First limited access release for r0p0	Revisions history
Updated manual structure and reorganized registers	Entire manual
Added direct RAM access information	<a href="#">10 Direct access to internal memory</a> on page 68
Updated system register descriptions	<a href="#">A AArch64 system registers</a> on page 132
Added memory-mapped registers	<a href="#">B External registers</a> on page 389

**Table C-4: Differences between Issue 0000-03 and Issue 0100-04**

Change	Location
First limited access release for r1p0	Revisions history
Updated manual structure and reorganized registers	Entire manual
Editorial changes	Entire manual

Change	Location
Updated number of instruction TLB and L1 data TLB entries	<a href="#">6.1 Memory Management Unit components</a> on page 49
Added more details on external aborts	<a href="#">6.6 Responses</a> on page 53
Updated MOP cache operations	<a href="#">7 L1 instruction memory system</a> on page 58
<ul style="list-style-type: none"> <li>Added BIM RAM returned data section</li> <li>Updated L1 data TLB returned data tables</li> </ul>	<a href="#">10 Direct access to internal memory</a> on page 68
Added victim location encoding	<a href="#">10.2 L2 cache encodings</a> on page 79
Updated system register descriptions	<a href="#">A AArch64 system registers</a> on page 132
Updated memory-mapped registers	<a href="#">B External registers</a> on page 389

**Table C-5: Differences between Issue 0100-04 and Issue 0200-05**

Change	Location
First early access release for r2p0	Revisions history
Editorial changes	Entire manual
Updated performance monitoring unit section with additional details on configuration options	<a href="#">3.1 Core components</a> on page 31
Updated for clarity	<a href="#">5.1 Voltage and power domains</a> on page 37
Added more complete details on warm reset mode	<a href="#">5.4.6 Warm reset mode</a> on page 45
Added more details on external aborts	<a href="#">6.6 Responses</a> on page 53
Updated tables	<a href="#">10.2 L2 cache encodings</a> on page 79
Added more complete details on fault detection and reporting	<a href="#">11.3 Fault detection and reporting</a> on page 88
Added coresight component Identification	<a href="#">16.6 CoreSight component identification</a> on page 102
Updated system register descriptions	<a href="#">A AArch64 system registers</a> on page 132
Updated memory-mapped registers	<a href="#">B External registers</a> on page 389

**Table C-6: Differences between Issue 0200-05 and Issue 0200-06**

Change	Location
Second early access release for r2p0	Revisions history
Editorial changes	Entire manual
Progressive terminology information added.	<a href="#">Release Information</a> on page 2
Note on cache indices removed	<a href="#">2.2 Cortex-X2 core configuration options</a> on page 24
Section on DSU dependent features added	<a href="#">2.3 DSU-110 dependent features</a> on page 24
Redundant bullet point removed	<a href="#">5.2.1 Wait for Interrupt and Wait for Event</a> on page 39
Additional information added in caution note	<a href="#">5.4 Core power modes</a> on page 41
Additional information added to Memory behavior and supported memory types section	<a href="#">6.7 Memory behavior and supported memory types</a> on page 55
Note added	<a href="#">7.1 L1 instruction cache behavior</a> on page 58
L1 data memory system features table updated	<a href="#">8 L1 data memory system</a> on page 61
Write streaming mode section added	<a href="#">8.5 Write streaming mode</a> on page 64
Debug chapter reorganized for clarity	<a href="#">16 Debug</a> on page 97
DS-5 corrected to Arm Debugger	<a href="#">16 Debug</a> on page 97
Bit fields updated	<a href="#">10.1.10 L1 data TLB returned data</a> on page 77



Change	Location
Core interfaces additional description added to ELA registers	<a href="#">16.2.1 Core interfaces</a> on page 99
Breakpoints and watchpoints topic updated	<a href="#">16.2.4 Breakpoints and watchpoints</a> on page 101
Updated system register and register descriptions	<a href="#">A AArch64 system registers</a> on page 132
<ul style="list-style-type: none"> <li>Reset value corrected for PF_MODE</li> <li>Descriptions corrected for TXREQ_LIMIT_DEC and TXREQ_LIMIT_INC</li> </ul>	<a href="#">A.1.16 IMP_CPUECTLR2_EL1, CPU Extended Control Register 2</a> on page 168
Updated memory-mapped registers	<a href="#">B External registers</a> on page 389

**Table C-7: Differences between Issue 0200-06 and Issue 0201-07**

Change	Location
First release for r2p1	Revisions history
Edits to DSU-110 dependent features section	<a href="#">2.3 DSU-110 dependent features</a> on page 24
Clarified core powerup and powerdown sequence	<a href="#">5.5 Cortex-X2 core powerup and powerdown sequence</a> on page 45
Added section on Page-based hardware attributes	<a href="#">6.8 Page-based hardware attributes</a> on page 56
Updated ERXADDR_EL1, Selected Error Record Address Register	<a href="#">A.10.6 ERXADDR_EL1, Selected Error Record Address Register</a> on page 369
Updated ARRAY in ERXMISCO_EL1, Selected Error Record Miscellaneous Register 0	<a href="#">A.10.10 ERXMISCO_EL1, Selected Error Record Miscellaneous Register 0</a> on page 378
Minor editorial change to registers PMEVCNTR6 through to PMEVCNTR19	<a href="#">B External registers</a> on page 389