Arm® Cortex®-A510 Core

Revision: r0p3

Technical Reference Manual

Non-Confidential

Issue 16
101604_0003_16_en

Arm® Cortex®-A510 Core

Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-01	30 August 2019	Confidential	First alpha release for r0p0
0000-02	20 December 2019	Confidential	First beta release for r0p0
0000-08	17 July 2020	Confidential	First limited access release for r0p0
0001-10	23 October 2020	Confidential	First early access release for r0p1
0002-11	11 December 2020	Confidential	First early access release for r0p2
0003-13	3 February 2021	Confidential	First early access release for r0p3
0003-16	25 May 2021	Non-Confidential	Second early access release for r0p3

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.

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.

Web Address

developer.arm.com

Progressive terminology 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.

Contents

1 Introduction....18
1.1 Product revision status....18
1.2 Intended audience....18
1.3 Conventions....18
1.4 Additional reading....20
1.5 Feedback....21

2 The Cortex®-A510 core....22
2.1 Cortex®-A510 core features....23
2.2 Cortex®-A510 core implementation options....24
2.3 DSU-110 dependent features....25
2.4 Supported standards and specifications....26
2.5 Test features....29
2.6 Design tasks....30
2.7 Product revisions....31

3 Technical overview....32
3.1 Core Components....32
3.2 Interfaces....36
3.3 Programmers model....37

4 Clocks and resets....38

5 Power management....39
5.1 Voltage and power domains....39
5.2 Architectural clock gating modes....43
5.2.1 Wait for Interrupt and Wait for Event....43
5.2.2 Low-power state behavior considerations....44
5.3 Power control....45
5.3.1 Maximum Power Mitigation Mechanism....45
5.4 Core power modes....46
5.4.1 On mode....48
5.4.2 Off mode....48

5.4.3 Emulated off mode........48
5.4.4 Functional retention mode........48
5.4.5 Full retention mode........49
5.4.6 Debug recovery mode........49
5.4.7 Warm reset mode........50
5.5 Complex power modes........50
5.6 Cortex®-A510 core powerup and powerdown sequence........53
5.7 Debug over powerdown........53

6 Memory management........54
6.1 Memory Management Unit components........54
6.2 Translation Lookaside Buffer entry content........56
6.3 Translation Lookaside Buffer match process........56
6.4 Translation table walks........57
6.5 Hardware management of the Access flag and dirty state........58
6.6 Responses........59
6.7 Memory behavior and supported memory types........60

7 L1 instruction memory system........62
7.1 L1 instruction cache behavior........62
7.2 L1 instruction cache Speculative memory access........63
7.3 Program flow prediction........63

8 L1 data memory system........65
8.1 L1 data cache behavior........65
8.2 Write streaming mode........66
8.3 Memory system implementation........67
8.4 Internal exclusive monitor........69
8.5 Data prefetching........70

9 L2 memory system........72
9.1 Optional integrated L2 cache........73
9.2 Support for memory types........73
9.3 Transaction capabilities........74

10 Direct access to internal memory........76
10.1 L1 cache encodings........77
10.2 L2 cache encodings........77

10.3 L2 TLB encodings........ 78

11 RAS extension support........ 79

11.1 Cache protection behavior........ 80

11.2 Error containment........ 81

11.3 Fault detection and reporting........ 81

11.4 Error detection and reporting........ 82

11.5 Error injection........ 83

11.6 RAS registers 1........ 83

11.7 RAS registers 2........ 84

12 GIC CPU interface........ 85

12.1 Disable the GIC CPU interface........ 85

12.2 GIC register summary........ 86

13 Advanced SIMD and floating-point support........ 87

14 Scalable Vector Extensions support........ 88

15 System control........ 89

15.1 Generic system control register summary........ 89

16 Debug........ 91

16.1 Supported debug methods........ 92

16.2 Debug register interfaces........ 93

16.2.1 Core interfaces........ 93

16.2.2 Effects of resets on Debug registers........ 94

16.2.3 External access permissions to Debug registers........ 95

16.2.4 Breakpoints and watchpoints........ 95

16.3 Debug events........ 95

16.4 Debug memory map and debug signals........ 96

16.5 ROM table........ 96

16.6 CoreSight component identification........ 97

16.7 ROM table register summary........ 97

16.8 Debug register summary........ 98

17 Performance Monitors Extension support........ 99

17.1 Performance monitors events........ 99

17.1.1 Architectural performance monitors events....99
17.1.2 Arm recommended implementation defined performance monitors events....108
17.1.3 implementation defined performance monitors events....111
17.2 Performance monitors interrupts....113
17.3 External register access permissions....114
17.4 Performance monitors register summary....114
17.5 Memory-mapped PMU register summary....114

18 Embedded Trace Extension support....116
18.1 Trace unit resources....117
18.2 Trace unit generation options....117
18.3 Reset the Embedded Trace Macrocell....118
18.4 Program and read the Embedded Trace Macrocell registers....119
18.5 Embedded Trace Macrocell register interfaces....121
18.6 Interaction with the Performance Monitoring Unit and Debug....121
18.7 Embedded Trace Extension events....122
18.8 ETE register summary....122

19 Trace Buffer Extension support....124
19.1 Program and read the trace buffer registers....124
19.2 Trace buffer register interface....124
19.3 TRBE register summary....124

20 Activity Monitors Extension support....125
20.1 Activity monitors access....125
20.2 Activity monitors counters....126
20.3 Activity monitors events....126
20.4 Activity monitors register summary....127
20.5 Memory-mapped AMU register summary....128

A IMPLEMENTATION DEFINED behaviors....129

B AArch64 registers....130
B.1 Generic system control register summary....130
B.1.1 AIDR_EL1, Auxiliary ID Register....131
B.1.2 ACTLR_EL1, Auxiliary Control Register (EL1)....132
B.1.3 ACTLR_EL2, Auxiliary Control Register (EL2)....134
B.1.4 HACR_EL2, Hypervisor Auxiliary Control Register....136

B.1.5 ACTLR_EL3, Auxiliary Control Register (EL3)....138
B.1.6 AMAIR_EL2, Auxiliary Memory Attribute Indirection Register (EL2)....140
B.1.7 LORID_EL1, LORegionID (EL1)....142
B.1.8 AMAIR_EL1, Auxiliary Memory Attribute Indirection Register (EL1)....143
B.1.9 AMAIR_EL3, Auxiliary Memory Attribute Indirection Register (EL3)....146
B.1.10 IMP_CPUACTLR_EL1, CPU Auxiliary Control Register....147
B.1.11 IMP_CPUACTLR2_EL1, CPU Auxiliary Control Register 2....149
B.1.12 IMP_CMPXACTLR_EL1, Complex Auxiliary Control Register....150
B.1.13 IMP_CPUECTLR_EL1, CPU Extended Control Register....152
B.1.14 IMP_CMPXECTLR_EL1, Complex Extended Control Register....157
B.1.15 IMP_CPUPWRCTLR_EL1, CPU Power Control Register....161
B.1.16 IMP_ATCR_EL2, CPU Auxiliary Translation Control Register....164
B.1.17 IMP_AVTCR_EL2, CPU Auxiliary Translation Control Register....166
B.1.18 IMP_ATCR_EL1, CPU Auxiliary Translation Control Register....169
B.1.19 IMP_ATCR_EL3, CPU Auxiliary Translation Control Register....172
B.1.20 AFSR0_EL2, Auxiliary Fault Status Register 0 (EL2)....173
B.1.21 AFSR1_EL2, Auxiliary Fault Status Register 1 (EL2)....175
B.1.22 AFSR0_EL1, Auxiliary Fault Status Register 0 (EL1)....178
B.1.23 AFSR1_EL1, Auxiliary Fault Status Register 1 (EL1)....180
B.1.24 AFSR0_EL3, Auxiliary Fault Status Register 0 (EL3)....182
B.1.25 AFSR1_EL3, Auxiliary Fault Status Register 1 (EL3)....184
B.2 Special purpose register summary....185
B.2.1 IMP_CPUPPMCR_EL3, Global PPM Configuration Register....185
B.3 Debug register summary....187
B.3.1 IMP_CDBGDR0_EL3, Cache Debug Data Register 0....187
B.4 System instruction register summary....198
B.4.1 SYS IMP_CDBGL1DCTR, L1 Data Cache Tag Read Operation....198
B.4.2 SYS IMP_CDBGL1ICTR, L1 Instruction Cache Tag Read Operation....199
B.4.3 SYS IMP_CDBGL2TR0, L2 TLB Read Operation 0....201
B.4.4 SYS IMP_CDBGL2CTR, L2 Cache Tag Read Operation....202
B.4.5 SYS IMP_CDBGL1DCDTR, L1 Data Cache Dirty Read Operation....203
B.4.6 SYS IMP_CDBGL1DCMR, L1 Data Cache MTE Tag Read Operation....205
B.4.7 SYS IMP_CDBGL2TR1, L2 TLB Read Operation 1....206
B.4.8 SYS IMP_CDBGL2CMR, L2 Cache MTE Tag Read Operation....207
B.4.9 SYS IMP_CDBGL1DCDR, L1 Data Cache Data Read Operation....208
B.4.10 SYS IMP_CDBGL1ICDR, L1 Instruction Cache Data Read Operation....210

B.4.11 SYS IMP_CDBGL2TR2, L2 TLB Read Operation 2....211
B.4.12 SYS IMP_CDBGL2CDR, L2 Cache Data Read Operation....212
B.5 Identification register summary....214
B.5.1 MIDR_EL1, Main ID Register....215
B.5.2 MPIDR_EL1, Multiprocessor Affinity Register....217
B.5.3 REVIDR_EL1, Revision ID Register....219
B.5.4 ID_PFR0_EL1, AArch32 Processor Feature Register 0....220
B.5.5 ID_PFR1_EL1, AArch32 Processor Feature Register 1....221
B.5.6 ID_DFR0_EL1, AArch32 Debug Feature Register 0....222
B.5.7 ID_AFR0_EL1, AArch32 Auxiliary Feature Register 0....223
B.5.8 ID_MMFR0_EL1, AArch32 Memory Model Feature Register 0....225
B.5.9 ID_MMFR1_EL1, AArch32 Memory Model Feature Register 1....226
B.5.10 ID_MMFR2_EL1, AArch32 Memory Model Feature Register 2....227
B.5.11 ID_MMFR3_EL1, AArch32 Memory Model Feature Register 3....229
B.5.12 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0....230
B.5.13 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1....231
B.5.14 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2....232
B.5.15 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3....234
B.5.16 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4....235
B.5.17 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5....236
B.5.18 ID_MMFR4_EL1, AArch32 Memory Model Feature Register 4....238
B.5.19 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6....239
B.5.20 MVFR0_EL1, AArch32 Media and VFP Feature Register 0....240
B.5.21 MVFR1_EL1, AArch32 Media and VFP Feature Register 1....241
B.5.22 MVFR2_EL1, AArch32 Media and VFP Feature Register 2....243
B.5.23 ID_PFR2_EL1, AArch32 Processor Feature Register 2....244
B.5.24 ID_AA64PFR0_EL1, AArch64 Processor Feature Register 0....245
B.5.25 ID_AA64PFR1_EL1, AArch64 Processor Feature Register 1....248
B.5.26 ID_AA64ZFR0_EL1, SVE Feature ID register 0....250
B.5.27 ID_AA64DFR0_EL1, AArch64 Debug Feature Register 0....252
B.5.28 ID_AA64DFR1_EL1, AArch64 Debug Feature Register 1....254
B.5.29 ID_AA64AFR0_EL1, AArch64 Auxiliary Feature Register 0....255
B.5.30 ID_AA64AFR1_EL1, AArch64 Auxiliary Feature Register 1....256
B.5.31 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0....258
B.5.32 ID_AA64ISAR1_EL1, AArch64 Instruction Set Attribute Register 1....261
B.5.33 ID_AA64MMFR0_EL1, AArch64 Memory Model Feature Register 0....263

B.5.34 ID_AA64MMFR1_EL1, AArch64 Memory Model Feature Register 1 265
B.5.35 ID_AA64MMFR2_EL1, AArch64 Memory Model Feature Register 2 267
B.5.36 CCSIDR_EL1, Current Cache Size ID Register 270
B.5.37 CLIDR_EL1, Cache Level ID Register 271
B.5.38 GMID_EL1, Multiple tag transfer ID register 275
B.5.39 CSSELR_EL1, Cache Size Selection Register 277
B.5.40 CTR_EL0, Cache Type Register 279
B.5.41 DCZID_EL0, Data Cache Zero ID register 281
B.5.42 MPAMIDR_EL1, MPAM ID Register (EL1) 282
B.5.43 IMP_CPUCFR_EL1, CPU Configuration Register 284
B.5.44 IMP_CPUMPMMCR_EL3, Global MPMM Configuration Register 286
B.6 Performance monitors register summary 288
B.6.1 PMMIR_EL1, Performance Monitors Machine Identification Register 288
B.6.2 PMCR_EL0, Performance Monitors Control Register 290
B.6.3 PMCEID0_EL0, Performance Monitors Common Event Identification register 0 294
B.6.4 PMCEID1_EL0, Performance Monitors Common Event Identification register 1 301
B.7 GIC register summary 308
B.7.1 ICC_CTLR_EL1, Interrupt Controller Control Register (EL1) 309
B.7.2 ICV_CTLR_EL1, Interrupt Controller Virtual Control Register 312
B.7.3 ICC_AP0R0_EL1, Interrupt Controller Active Priorities Group 0 Registers 315
B.7.4 ICV_AP0R0_EL1, Interrupt Controller Virtual Active Priorities Group 0 Registers 316
B.7.5 ICC_AP1R0_EL1, Interrupt Controller Active Priorities Group 1 Registers 318
B.7.6 ICV_AP1R0_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers 319
B.7.7 ICH_VTR_EL2, Interrupt Controller VGIC Type Register 320
B.7.8 ICC_CTLR_EL3, Interrupt Controller Control Register (EL3) 322
B.8 Activity monitors register summary 326
B.8.1 AMEVTYPER10_EL0, Activity Monitors Event Type Registers 1 327
B.8.2 AMEVTYPER11_EL0, Activity Monitors Event Type Registers 1 328
B.8.3 AMEVTYPER12_EL0, Activity Monitors Event Type Registers 1 329
B.8.4 AMCFGR_EL0, Activity Monitors Configuration Register 330
B.8.5 AMCGCR_EL0, Activity Monitors Counter Group Configuration Register 332
B.8.6 AMEVTYPER00_EL0, Activity Monitors Event Type Registers 0 333
B.8.7 AMEVTYPER01_EL0, Activity Monitors Event Type Registers 0 334
B.8.8 AMEVTYPER02_EL0, Activity Monitors Event Type Registers 0 335
B.8.9 AMEVTYPER03_EL0, Activity Monitors Event Type Registers 0 336
B.9 RAS register summary 337

B.9.1 ERRIDR_EL1, Error Record ID Register....337
B.9.2 ERRSELR_EL1, Error Record Select Register.... 339
B.10 Trace register summary....341
B.10.1 TRCIDR8, ID Register 8.... 341
B.10.2 TRCIMSPEC0, IMP DEF Register 0.... 343
B.10.3 TRCIDR9, ID Register 9.... 344
B.10.4 TRCIDR2, ID Register 2.... 346
B.10.5 TRCIDR3, ID Register 3.... 347
B.10.6 TRCIDR4, ID Register 4.... 350
B.10.7 TRCIDR5, ID Register 5.... 352
B.10.8 TRCIDR6, ID Register 6.... 354
B.10.9 TRCIDR7, ID Register 7.... 355
B.10.10 TRCIDR10, ID Register 10....356
B.10.11 TRCIDR11, ID Register 11....358
B.10.12 TRCIDR12, ID Register 12....359
B.10.13 TRCIDR13, ID Register 13....360
B.10.14 TRCAUXCTLR, Auxiliary Control Register.... 361
B.10.15 TRCIDR0, ID Register 0....363
B.10.16 TRCIDR1, ID Register 1....366
B.10.17 TRCDEVARCH, Device Architecture Register.... 367
B.10.18 TRCDEVID, Device Configuration Register....369
B.10.19 TRCCLAIMSET, Claim Tag Set Register....370
B.10.20 TRCCLAIMCLR, Claim Tag Clear Register.... 373
B.11 TRBE register summary.... 376
B.11.1 TRBIDR_EL1, Trace Buffer ID Register....376

C External registers....379
C.1 MPMM register summary....379
C.1.1 CPUPPMCR, Global PPM Configuration Register.... 379
C.1.2 CPUMPMMCR, Global MPMM Configuration Register....380
C.2 Memory-mapped PMU register summary.... 381
C.2.1 PMPCSSR, Snapshot Program Counter Sample Register....382
C.2.2 PMCIDSSR, Snapshot CONTEXTIDR_EL1 Sample Register.... 383
C.2.3 PMCID2SSR, Snapshot CONTEXTIDR_EL2 Sample Register....384
C.2.4 PMSSSR, PMU Snapshot Status Register....385
C.2.5 PMOVSSR, PMU Overflow Status Snapshot Register....386

C.2.6 PMCCNTSR, PMU Cycle Counter Snapshot Register....387
C.2.7 PMEVCNTSR0, PMU Event Counter Snapshot Register....387
C.2.8 PMEVCNTSR1, PMU Event Counter Snapshot Register....388
C.2.9 PMEVCNTSR2, PMU Event Counter Snapshot Register....389
C.2.10 PMEVCNTSR3, PMU Event Counter Snapshot Register....390
C.2.11 PMEVCNTSR4, PMU Event Counter Snapshot Register....390
C.2.12 PMEVCNTSR5, PMU Event Counter Snapshot Register....391
C.2.13 PMSSCR, PMU Snapshot Capture Register....392
C.2.14 PMCFGR, Performance Monitors Configuration Register....393
C.2.15 PMCR_EL0, Performance Monitors Control Register....394
C.2.16 PMCEID0, Performance Monitors Common Event Identification register 0....397
C.2.17 PMCEID1, Performance Monitors Common Event Identification register 1....401
C.2.18 PMCEID2, Performance Monitors Common Event Identification register 2....405
C.2.19 PMCEID3, Performance Monitors Common Event Identification register 3....409
C.2.20 PMMIR, Performance Monitors Machine Identification Register....412
C.2.21 PMDEVARCH, Performance Monitors Device Architecture register....413
C.2.22 PMDEVID, Performance Monitors Device ID register....414
C.2.23 PMDEVTYPE, Performance Monitors Device Type register....415
C.2.24 PMPIDR4, Performance Monitors Peripheral Identification Register 4....416
C.2.25 PMPIDR0, Performance Monitors Peripheral Identification Register 0....417
C.2.26 PMPIDR1, Performance Monitors Peripheral Identification Register 1....418
C.2.27 PMPIDR2, Performance Monitors Peripheral Identification Register 2....419
C.2.28 PMPIDR3, Performance Monitors Peripheral Identification Register 3....420
C.2.29 PMCIDR0, Performance Monitors Component Identification Register 0....421
C.2.30 PMCIDR1, Performance Monitors Component Identification Register 1....421
C.2.31 PMCIDR2, Performance Monitors Component Identification Register 2....422
C.2.32 PMCIDR3, Performance Monitors Component Identification Register 3....423
C.3 Debug register summary....424
C.3.1 EDRCR, External Debug Reserve Control Register....424
C.3.2 EDACR, External Debug Auxiliary Control Register....426
C.3.3 EDPRCR, External Debug Power/Reset Control Register....426
C.3.4 MIDR_EL1, Main ID Register....428
C.3.5 EDPFR, External Debug Processor Feature Register....429
C.3.6 EDDFR, External Debug Feature Register....431
C.3.7 EDDEVARCH, External Debug Device Architecture register....432
C.3.8 EDDEVID2, External Debug Device ID register 2....433

C.3.9 EDDEVID1, External Debug Device ID register 1......434
C.3.10 EDDEVID, External Debug Device ID register 0......435
C.3.11 EDDEVTYPE, External Debug Device Type register......436
C.3.12 EDPIDR4, External Debug Peripheral Identification Register 4......437
C.3.13 EDPIDR0, External Debug Peripheral Identification Register 0......437
C.3.14 EDPIDR1, External Debug Peripheral Identification Register 1......438
C.3.15 EDPIDR2, External Debug Peripheral Identification Register 2......439
C.3.16 EDPIDR3, External Debug Peripheral Identification Register 3......440
C.3.17 EDCIDR0, External Debug Component Identification Register 0......441
C.3.18 EDCIDR1, External Debug Component Identification Register 1......442
C.3.19 EDCIDR2, External Debug Component Identification Register 2......443
C.3.20 EDCIDR3, External Debug Component Identification Register 3......444
C.4 Memory-mapped AMU register summary......444
C.4.1 AMEVTYPER00, Activity Monitors Event Type Registers 0......445
C.4.2 AMEVTYPER01, Activity Monitors Event Type Registers 0......446
C.4.3 AMEVTYPER02, Activity Monitors Event Type Registers 0......447
C.4.4 AMEVTYPER03, Activity Monitors Event Type Registers 0......448
C.4.5 AMEVTYPER10, Activity Monitors Event Type Registers 1......449
C.4.6 AMEVTYPER11, Activity Monitors Event Type Registers 1......449
C.4.7 AMEVTYPER12, Activity Monitors Event Type Registers 1......450
C.4.8 AMCGCR, Activity Monitors Counter Group Configuration Register......451
C.4.9 AMCFGR, Activity Monitors Configuration Register......452
C.4.10 AMIIDR, Activity Monitors Implementation Identification Register......453
C.4.11 AMDEVARCH, Activity Monitors Device Architecture Register......454
C.4.12 AMDEVTYPE, Activity Monitors Device Type Register......455
C.4.13 AMPIDR4, Activity Monitors Peripheral Identification Register 4......456
C.4.14 AMPIDR0, Activity Monitors Peripheral Identification Register 0......457
C.4.15 AMPIDR1, Activity Monitors Peripheral Identification Register 1......458
C.4.16 AMPIDR2, Activity Monitors Peripheral Identification Register 2......459
C.4.17 AMPIDR3, Activity Monitors Peripheral Identification Register 3......460
C.4.18 AMCIDR0, Activity Monitors Component Identification Register 0......461
C.4.19 AMCIDR1, Activity Monitors Component Identification Register 1......462
C.4.20 AMCIDR2, Activity Monitors Component Identification Register 2......463
C.4.21 AMCIDR3, Activity Monitors Component Identification Register 3......463
C.5 RAS register summary......464
C.5.1 ERR2CTLR, Error Record Control Register......465

C.5.2 ERR1CTLR, Error Record Control Register....... 468
C.5.3 ERR2FR, Error Record Feature Register....... 470
C.5.4 ERR1FR, Error Record Feature Register....... 473
C.5.5 ERR2MISC1, Error Record Miscellaneous Register 1....... 476
C.5.6 ERR1MISC1, Error Record Miscellaneous Register 1....... 477
C.5.7 ERR2MISC0, Error Record Miscellaneous Register 0....... 477
C.5.8 ERR1MISC0, Error Record Miscellaneous Register 0....... 479
C.5.9 ERR2PFGF, Pseudo-fault Generation Feature Register....... 481
C.5.10 ERR1PFGF, Pseudo-fault Generation Feature Register....... 484
C.5.11 ERR2PFGCTL, Pseudo-fault Generation Control Register....... 486
C.5.12 ERR1PFGCTL, Pseudo-fault Generation Control Register....... 489
C.5.13 ERR2STATUS, Error Record Primary Status Register....... 493
C.5.14 ERR1STATUS, Error Record Primary Status Register....... 499
C.5.15 ERR2MISC3, Error Record Miscellaneous Register 3....... 505
C.5.16 ERR1MISC3, Error Record Miscellaneous Register 3....... 506
C.5.17 ERR2MISC2, Error Record Miscellaneous Register 2....... 507
C.5.18 ERR1MISC2, Error Record Miscellaneous Register 2....... 508
C.6 ETE register summary....... 509
C.6.1 TRCAUXCTLR, Auxiliary Control Register....... 510
C.6.2 TRCIDR8, ID Register 8....... 511
C.6.3 TRCIDR9, ID Register 9....... 511
C.6.4 TRCIDR10, ID Register 10....... 512
C.6.5 TRCIDR11, ID Register 11....... 513
C.6.6 TRCIDR12, ID Register 12....... 514
C.6.7 TRCIDR13, ID Register 13....... 514
C.6.8 TRCIMSPEC0, IMP DEF Register 0....... 515
C.6.9 TRCIDR0, ID Register 0....... 516
C.6.10 TRCIDR1, ID Register 1....... 518
C.6.11 TRCIDR2, ID Register 2....... 519
C.6.12 TRCIDR3, ID Register 3....... 520
C.6.13 TRCIDR4, ID Register 4....... 522
C.6.14 TRCIDR5, ID Register 5....... 523
C.6.15 TRCIDR6, ID Register 6....... 524
C.6.16 TRCIDR7, ID Register 7....... 525
C.6.17 TRCITCTRL, Integration Mode Control Register....... 526
C.6.18 TRCCLAIMSET, Claim Tag Set Register....... 527

C.6.19 TRCCLAIMCLR, Claim Tag Clear Register........ 528
C.6.20 TRCDEVARCH, Device Architecture Register........ 530
C.6.21 TRCDEVID2, Device Configuration Register 2........531
C.6.22 TRCDEVID1, Device Configuration Register 1........532
C.6.23 TRCDEVID, Device Configuration Register........533
C.6.24 TRCDEVTYPE, Device Type Register........ 534
C.6.25 TRCPIDR4, Peripheral Identification Register 4........ 535
C.6.26 TRCPIDR5, Peripheral Identification Register 5........ 536
C.6.27 TRCPIDR6, Peripheral Identification Register 6........ 537
C.6.28 TRCPIDR7, Peripheral Identification Register 7........ 537
C.6.29 TRCPIDR0, Peripheral Identification Register 0........ 538
C.6.30 TRCPIDR1, Peripheral Identification Register 1........ 539
C.6.31 TRCPIDR2, Peripheral Identification Register 2........ 540
C.6.32 TRCPIDR3, Peripheral Identification Register 3........ 541
C.6.33 TRCCIDR0, Component Identification Register 0........542
C.6.34 TRCCIDR1, Component Identification Register 1........543
C.6.35 TRCCIDR2, Component Identification Register 2........544
C.6.36 TRCCIDR3, Component Identification Register 3........545
C.7 ROM table register summary........ 546
C.7.1 ROMENTRY0, Class 0x9 ROM Table Entries........547
C.7.2 ROMENTRY1, Class 0x9 ROM Table Entries........548
C.7.3 ROMENTRY2, Class 0x9 ROM Table Entries........550
C.7.4 ROMENTRY3, Class 0x9 ROM Table Entries........551
C.7.5 ROMENTRY4, Class 0x9 ROM Table Entries........553
C.7.6 ROMENTRY5, Class 0x9 ROM Table Entries........555
C.7.7 ROMENTRY6, Class 0x9 ROM Table Entries........556
C.7.8 ROMENTRY7, Class 0x9 ROM Table Entries........558
C.7.9 DEVARCH, Device Architecture Register........559
C.7.10 DEVID2, Device Configuration Register 2........ 560
C.7.11 DEVID1, Device Configuration Register 1........ 561
C.7.12 DEVID, Device Configuration Register........562
C.7.13 DEVTYPE, Device Type Register........ 563
C.7.14 PIDR4, Peripheral Identification Register 4........ 564
C.7.15 PIDR5, Peripheral Identification Register 5........ 564
C.7.16 PIDR6, Peripheral Identification Register 6........ 565
C.7.17 PIDR7, Peripheral Identification Register 7........ 566

C.7.18 PIDR0, Peripheral Identification Register 0 567
C.7.19 PIDR1, Peripheral Identification Register 1 567
C.7.20 PIDR2, Peripheral Identification Register 2 568
C.7.21 PIDR3, Peripheral Identification Register 3 569
C.7.22 CIDR0, Component Identification Register 0 570
C.7.23 CIDR1, Component Identification Register 1 571
C.7.24 CIDR2, Component Identification Register 2 571
C.7.25 CIDR3, Component Identification Register 3 572

D Document revisions 574
D.1 Revisions 574

1 Introduction

1.1 Product revision status

The r*x*p*y* identifier indicates the revision status of the product described in this manual, for example, r*1*p*2*, 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.

Typographic conventions

Convention	Use
italic	Introduces citations.
bold	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.
`monospace bold`	Denotes language keywords when used outside example code.
`monospace underline`	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: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`

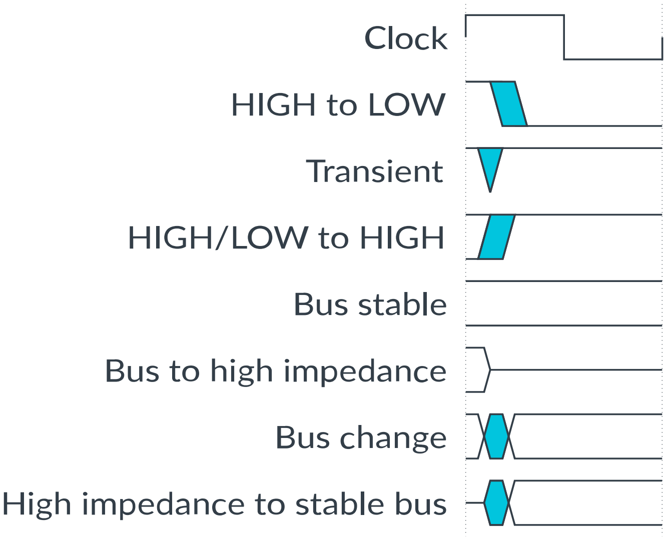
Convention	Use
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.
⚠ Caution	This represents a recommendation which, if not followed, 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®-A510 Release Note	-	Yes
Arm® Cortex®-A510 Core Configuration and Integration Manual	101605	Yes
Arm® Cortex®-A510 Core Cryptographic Extension Technical Reference Manual	101606	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 Armv8-A architecture profile	DDI 0487	No
Arm® Architecture Reference Manual Supplement, The Scalable Vector Extension (SVE) for Armv8-A	DDI 0584	No
Arm® Reliability, Availability, and Serviceability (RAS) Specification Armv8, for the Armv8-A architecture profile	DDI 0587	No
Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM) for Armv8-A	DDI 0598	No
Arm®v9-A Supplement for v8-A Arm® Architecture Reference Manual	DDI 0608	No
Arm® CoreSight™ Architecture Specification v3.0	IHI 0029	No
AMBA® 5 CHI Architecture Specification	IHI 0050	No

Document Name	Document ID	Licensee only
Arm® Embedded Trace Macrocell Architecture Specification	IHI 0064	No
Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4	IHI 0069	No

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. Give:

- The title Arm® Cortex®-A510 Core Technical Reference Manual.
- The number 101604_0003_16_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.

Note

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®-A510 core

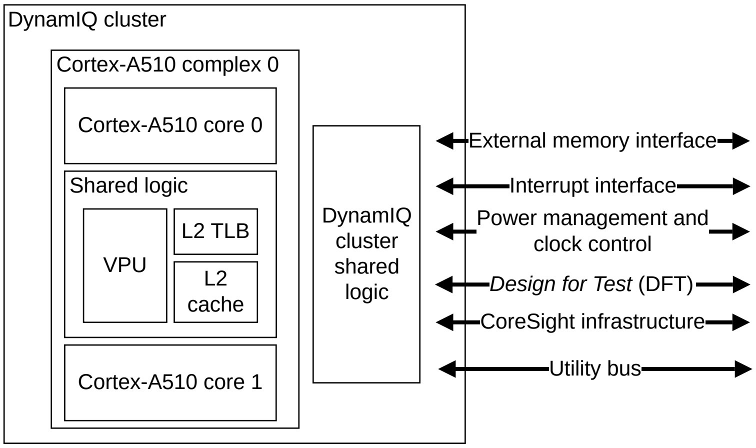
The Cortex®-A510 core is a high-efficiency, low-power product that implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.5-A.

The Cortex®-A510 core is implemented inside a DSU-110 DynamIQ™ cluster and is always connected to the *DynamIQ™ Shared Unit-110* (DSU-110). The DSU-110 behaves as a full interconnect with L3 cache and snoop control. This connection configuration is also used in systems with different types of cores where the Cortex®-A510 is the high efficiency core.

Cortex®-A510 cores are implemented inside a block called a complex, which contains up to two cores. Within a dual-core complex, the *Vector Processing Unit* (VPU), the L2 *Translation Lookaside Buffer* (TLB), and the L2 cache logic are shared between cores.

The following figure shows an example of a dual-core configuration:

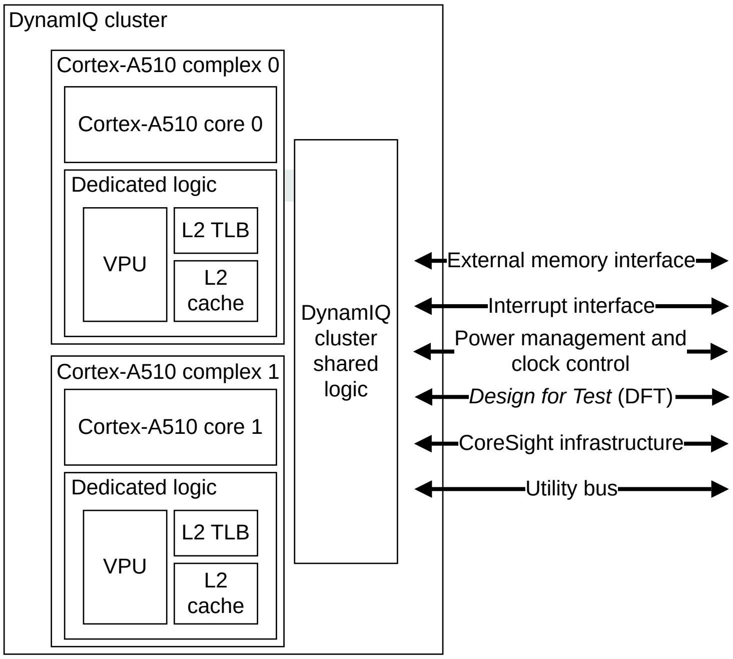
Figure 2-1: Example configuration with a Cortex®-A510 dual-core complex



You can also configure a complex that contains a single Cortex®-A510 core with dedicated logic. You can configure your systems so that all cores are configured in single-core complexes. This type of configuration improves performance but at the cost of area efficiency.

The following figure shows an example of a cluster with single-core complexes:

Figure 2-2: Example configuration with two Cortex®-A510 single-core complexes



Note

This manual applies to the Cortex®-A510 core only. Read this manual together with the *Arm® DynamIQ Shared Unit-110 Technical Reference Manual* for detailed information about the DSU-110.

2.1 Cortex®-A510 core features

The Cortex®-A510 core might be used in standalone DynamIQ™ configurations where a homogenous DSU-110 DynamIQ™ cluster includes one to eight Cortex®-A510 cores. The Cortex®-A510 core might also be used as a high efficiency core or a high-performance core in a heterogenous DSU-110 DynamIQ™ cluster.

However, regardless of the cluster configuration, the Cortex®-A510 core always has the same features.

Core features

- Implementation of the Arm®v9.0-A A64 instruction set
- AArch64 Execution state at all Exception levels, EL0 to EL3
- Separate L1 data and instruction side memory systems with a *Memory Management Unit* (MMU)
- In-order pipeline with direct and indirect branch prediction
- *Generic Interrupt Controller* (GIC) CPU interface to connect to an external interrupt distributor
- Generic Timer interface that supports a 64-bit count input from an external system counter

- Implementation of the *Reliability, Availability, and Serviceability* (RAS) Extension
- *Scalable Vector Extension* (SVE) and SVE2 SIMD instruction set, offering Advanced SIMD and floating-point architecture support
- Support for the optional Cryptographic Extension, which is licensed separately
- *Activity Monitoring Unit* (AMU)

Cache features

- Separate L1 data and instruction caches
- Optional unified L2 cache
- L1 and L2 cache protection with *Error Correcting Code* (ECC) or parity
- Support for *Memory system resource Partitioning And Monitoring* (MPAM)

Debug features

- Arm®v9.0-A debug logic
- *Performance Monitoring Unit* (PMU)
- *Embedded Trace Macrocell* (ETM) with support for *Embedded Trace Extension* (ETE)
- *TRace Buffer Extension* (TRBE)
- Optional *Embedded Logic Analyzer* (ELA)

Related information

3 Technical overview on page 32

2.2 Cortex®-A510 core implementation options

You can choose the options that fit your implementation needs at build-time configuration. These options apply to all cores in a complex, or to all cores in the DSU-110 DynamIQ™ cluster.

You can configure your Cortex®-A510 core implementation using the following options:

Dual or single core

You can group cores into dual-core complexes, or instantiate them as single-core complexes. Dual-core complexes share the L2 cache, the L2 *Translation Lookaside Buffer* (TLB), and the *Vector Processing Unit* (VPU), while single-core complexes have a dedicated L2 cache, L2 TLB, and VPU.

Cryptographic Extension

Configure your implementation with or without the Cryptographic Extension. The selected option applies to all cores in the DynamIQ™ cluster, including non-Cortex®-A510 cores. The Cryptographic Extension is an optional separately licensable product.

Vector datapath size

The size of the vector datapaths can be 2×64-bit or 2×128-bit. The selected option applies to all Cortex®-A510 cores in the DynamIQ™ cluster.

ECC or parity core cache protection

Configure whether your core implementation includes cache protection. The selected option applies to all cores in the DynamIQ™ cluster, including non-Cortex®-A510 cores.

CoreSight™ Embedded Logic Analyzer

Optionally, you can include support for integrating CoreSight™ ELA-600, as a separately licensable product.

L1 instruction cache size

The L1 instruction cache can be 32KB or 64KB. The selected option applies to all Cortex®-A510 cores in the DynamIQ™ cluster.

L1 data cache size

The L1 data cache can be 32KB or 64KB. The selected option applies to all Cortex®-A510 cores in the DynamIQ™ cluster.

L2 cache

Configure whether the L2 cache is present.

L2 cache size

The L2 cache size for the complex can be 128KB, 192KB, 256KB, 384KB, or 512KB.

L2 slices

The number of L2 cache slices can be one or two.

L2 cache data RAM partitions

The number of partitions in the L2 cache data RAMs can be one or two.

Evict/Allocate feature

Configure whether the *Evict/Allocate* (EVA) feature is used on the L2 cache data RAMs.

See *RTL configuration process* in the *Arm® Cortex®-A510 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®-A510 core.

Table 2-1: Cortex®-A510 core features that have a dependency on the DSU-110

Feature	Supported in the Cortex®-A510 core	Dependency on the DSU-110
Direct connect	No	Direct connect support at the DSU-110 DynamIQ™ cluster level only applies when your licensed core also supports Direct connect. Direct connect is intended for large systems where there are many cores.

Feature	Supported in the Cortex®-A510 core	Dependency on the DSU-110
Core included in a complex	Yes	Affects the DynamIQ™ cluster configuration and external signals.
Cryptographic Extension	Yes	Affects the external signals of the DSU-110.
Maximum Power Mitigation Mechanism (MPMM)	Yes	
Performance Defined Power (PDP) feature	No	
DISPBLKx signal supported	No	
Statistical Profiling Extension (SPE) architecture	No	

Note

The Cryptographic Extension is supplied under a separate license.

2.4 Supported standards and specifications

The Cortex®-A510 core implements the Arm®v9.0-A architecture. The Arm®v9.0-A architecture extends the architecture defined in the Arm®v8-A architectures up to Arm®v8.5-A. The core also implements specific Arm architecture extensions and implements interconnect, interrupt, timer, debug, and trace architectures.

The Cortex®-A510 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®-A510 core supports.

Table 2-2: Armv8.0-A optional feature support in the Cortex®-A510 core

Feature	Status	Notes
Cryptographic Extension	Supported, using a configurable option	See the *Arm® Cortex®-A510 Core Cryptographic Extension Technical Reference Manual* for more technical reference and register information. This extension is licensed separately and access to the documentation is restricted by contract with Arm.
Advanced SIMD and floating-point support	Supported	See 13 Advanced SIMD and floating-point support on page 87 for more technical reference and register information.
Performance Monitors Extension	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.
CP15SDISABLE2 input	Not supported	-

Table 2-3: Arm®v8.1-A optional feature support in the Cortex®-A510 core

Feature	Status	Notes
Armv8.1-TTHM, Hardware management of the Access flag and dirty state	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.1-VMID16, 16-bit VMID	Supported	
Enhanced PAN	Supported	Enhancement for *Privileged Access Never* (PAN) with Execute-only.

Table 2-4: Arm®v8.2-A optional feature support in the Cortex®-A510 core

Feature	Status	Notes
Armv8.2-TTPBHA, Translation Table Page-Based Hardware Attributes	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.2-PCSample, PC Sample-based profiling	Supported	
Armv8.2-SHA, SHA2-512 and SHA3 functionality	Supported as part of Armv8-A Cryptographic Extension	
Armv8.2-SM, SM3 and SM4 functionality	Supported as part of Armv8-A Cryptographic Extension	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.2-BF16, 16-bit floating-point instructions	Supported	
Armv8.2-I8MM, Int8 Matrix Multiply instructions	Supported	
Scalable Vector Extension (SVE)	Supported	See 14 Scalable Vector Extensions support on page 88 and the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on this extension.
Armv8.2-LPA, Large *Physical Address* (PA) and *Intermediate PA* (IPA) support	Not supported	-
Armv8.2-LVA, Large *Virtual Address* (VA) support	Not supported	-
Armv8.2-LSMAOC, Load/Store Multiple Atomicity and Ordering Controls	Not supported	-
Armv8.2-AA32HPD, AArch32 Hierarchical Permission Disables	Not supported	-
Statistical Profiling Extension (SPE)	Not supported	-

Table 2-5: Arm®v8.3-A optional feature support in the Cortex®-A510 core

Feature	Status	Notes
Armv8.3-NV, Nested Virtualization	Not supported	-
Armv8.3-CCIDX, Cache extended number of sets	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on these features.
Armv8.3-PAuth2, Pointer Authentication enhancements	Supported	

Feature	Status	Notes
Armv8.3-FPAC, *Faulting Pointer Authentication Code* (FPAC)	Supported	

Table 2-6: Arm®v8.4-A optional feature support in the Cortex®-A510 core

Feature	Status	Notes
Activity Monitors Extension	Supported	See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on this feature.
Memory system resource Partitioning And Monitoring (MPAM) Extension	Supported	See the *Arm® Architecture Reference Manual Supplement, Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A* for information on this extension.
Armv8.4-NV, enhanced support for Nested Virtualization	Not supported	-

Table 2-7: Arm®v8.5-A optional feature support in the Cortex®-A510 core

Feature	Status	Notes
Armv8.5-MemTag, *Memory Tagging Extension* (MTE)	Supported	The Cortex®-A510 core always implements MTE. See *CHI master interface* in the *Arm® DynamIQ Shared Unit-110 Technical Reference Manual* for information on CHI.E commands inferred by MTE.
Armv8.5-RNG, Random Number Generator instructions	Not supported	-
Armv8.5-CSEH, Context Synchronization and Exception Handling	Not supported	-

The following table shows the Arm®v9.0-A features that the Cortex®-A510 core supports.

Table 2-8: Arm®v9.0-A feature support in the Cortex®-A510 core

Feature	Status	Notes
Scalable Vector Extension 2 (SVE2)	Supported	See 14 Scalable Vector Extensions support on page 88.
Embedded Trace Extension (ETE)	Supported	See 18 Embedded Trace Extension support on page 116.
TRace Buffer Extension (TRBE)	Supported	See 19 Trace Buffer Extension support on page 124.
SVE2 SM4 instructions	Supported, using a configurable option	See the *Arm®v9-A Supplement for v8-A Arm® Architecture Reference Manual*
SVE2 SHA-3 instructions		
SVE2 bit permute instructions		
SVE2 AES instructions		
Transactional Memory Extension (TME)	Not supported	-

The following table shows the other standards and specifications that the Cortex®-A510 core supports.

Table 2-9: Other standards and specifications support in the Cortex®-A510 core

Standard or specification	Version	Notes
Generic Interrupt Controller (GIC)	GICv4.1	See the *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4* for more information.

Standard or specification	Version	Notes
Debug	-	Arm®v9.0-A architecture implemented with ARMv8.3-DoPD, Debug over powerdown and ARMv8.4-Debug, Debug relaxations and extensions support. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on this architecture.
CoreSight	v3.0	See the *Arm® CoreSight™ Architecture Specification v3.0* for more information.
Reliability, Availability, and Serviceability (RAS)	-	All extensions up to Arm®v9.0-A with *Error Correcting Code* (ECC) configured. See 11 RAS extension support on page 79 for more information on the implementation of this extension in the core.

Related information

3.1 Core Components on page 32

2.5 Test features

The Cortex®-A510 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®-A510 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®-A510 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.

Note

The *Arm® Cortex®-A510 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®-A510 core is delivered as a synthesizable RTL description in SystemVerilog. Before you can use the Cortex®-A510 core, you must implement, integrate, and program it.

Separate parties can perform each of the following tasks. Implementation and integration choices affect the behavior and features of the core:

Implementation

The implementer configures and synthesizes the RTL to produce a hard macrocell. This task includes integrating RAMs into the design.

Integration

The integrator connects the macrocell into a *System on Chip* (SoC). This task includes connecting it to a memory system and peripherals.

Programming

In the final task, the system programmer develops the software to configure and initialize the core and tests the application software.

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

Build configuration

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

Configuration inputs

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. They can also limit the options available to the software.

Software configuration

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

See *RTL configuration process* in the *Arm® Cortex®-A510 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.

Note

The *Arm® Cortex®-A510 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.

2.7 Product revisions

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

Table 2-10: Product revisions

Revision	Notes
r0p0	First release for r0p0
r0p1	Further development and optimization of the product, including addition of the *TRace Buffer Extension* (TRBE)
r0p2	Maintenance release, includes various performance improvements
r0p3	Maintenance release, includes various performance improvements

Changes in functionality that have an impact on the documentation also appear in D.1 Revisions on page 574.

3 Technical overview

The components in the Cortex®-A510 core are designed to make it a high efficiency core.

The components include:

- *Instruction Fetch Unit* (IFU)
- *Data Processing Unit* (DPU)
- L1 instruction and L1 data memory systems
- *Memory Management Unit* (MMU)
- *Embedded Trace Macrocell* (ETM)
- *TRace Buffer Extension* (TRBE)
- *Vector Processing Unit* (VPU)
- *Generic Interrupt Controller* (GIC) CPU interface
- L2 *Translation Lookaside Buffer* (TLB)
- L2 memory system with optional L2 cache
- Optional Cryptographic Extension
- Optional *Embedded Logic Analyzer* (ELA)

The Cortex®-A510 core interfaces with the *DynamIQ™ Shared Unit-110* (DSU-110) through the CPU bridge.

The programmers model and the architecture features that are implemented in the Cortex®-A510 core comply with the standards in 2.4 Supported standards and specifications on page 26.

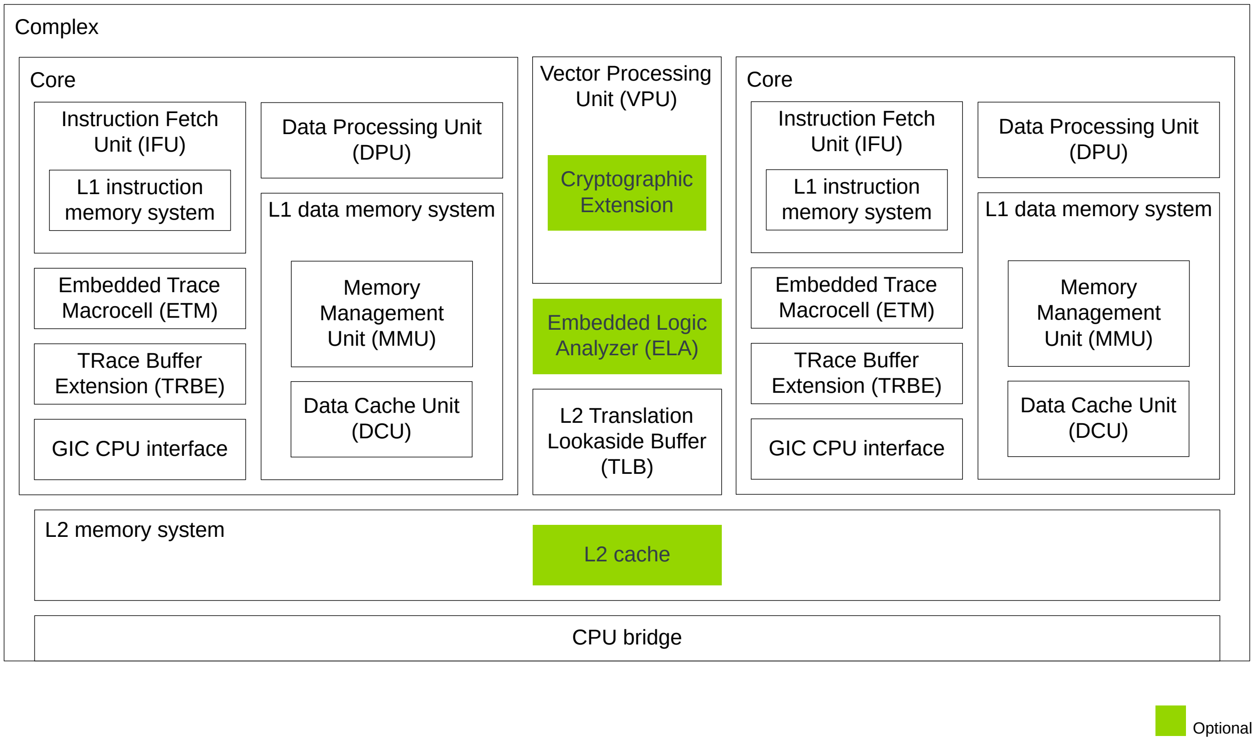
3.1 Core Components

The Cortex®-A510 core includes components that are designed to make it a high-efficiency, low-power, and area-efficient product.

Cortex®-A510 cores are always implemented inside a complex. A Cortex®-A510 complex includes a CPU bridge that connects the complex to the *DynamIQ™ Shared Unit-110* (DSU-110). The DSU-110 connects the complex to an external memory system and to the rest of the *System on Chip* (SoC).

The following figure shows the components within a Cortex®-A510 complex:

Figure 3-1: Cortex®-A510 core components


Complex
Core
Instruction Fetch Unit (IFU)
L1 instruction memory system
Data Processing Unit (DPU)
L1 data memory system
Embedded Trace Macrocell (ETM)
Memory Management Unit (MMU)
TRace Buffer Extension (TRBE)
Data Cache Unit (DCU)
GIC CPU interface
Vector Processing Unit (VPU)
Cryptographic Extension
Embedded Logic Analyzer (ELA)
L2 Translation Lookaside Buffer (TLB)
Core
Instruction Fetch Unit (IFU)
L1 instruction memory system
Data Processing Unit (DPU)
L1 data memory system
Embedded Trace Macrocell (ETM)
Memory Management Unit (MMU)
TRace Buffer Extension (TRBE)
Data Cache Unit (DCU)
GIC CPU interface
L2 memory system
L2 cache
CPU bridge
Optional

Instruction Fetch Unit

The IFU fetches instructions from the instruction cache or from external memory and uses a dynamic branch predictor to predict the outcome of branches in the instruction stream. It passes the instructions to the DPU for processing.

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 fully associative L1 instruction TLB
- A 32KB or 64KB 4-way set associative L1 instruction cache with 64-byte cache lines

Data Processing Unit

The DPU decodes and executes instructions. It executes instructions that require data transfer to or from the memory system by interfacing to the DCU. The DPU includes the *Performance Monitoring Unit* (PMU) and the *Activity Monitoring Unit* (AMU).

Performance Monitoring Unit

The PMU provides six 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.

Activity Monitoring Unit

The Cortex®-A510 core includes an AMU, which, like the PMU, counts certain events that are related to the behavior of the core. The AMU implements seven event counters. Activity monitoring is intended for system management use whereas performance monitoring is aimed at user and debug applications.

The AMU registers are accessible using the System registers or the DSU-110 DynamIQ™ cluster Utility bus.

L1 data memory system

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

The L1 data memory system includes:

- An MMU
- A fully associative L1 data TLB
- A 32KB or 64KB, 4-way set associative cache with 64-byte cache lines
- A DCU that handles load/store and System register access operations
- A *Bus Interface Unit* (BIU) that handles the linefills to the L1 data cache
- A *STore Buffer* (STB) that handles store instructions, cache and TLB maintenance operations, and barriers

The 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. The TLB stores these mappings when translating an address.

Embedded Trace Macrocell and Trace Buffer Extension

The Cortex®-A510 core supports a range of debug, test, and trace options including an ETM and TRBE.

The Cortex®-A510 core also includes a ROM table that contains a list of system components. Debuggers can use the ROM table to determine which CoreSight components are implemented.

All the debug and trace components of the Cortex®-A510 core are described in this manual. The *Arm® Cortex®-A510 Core Configuration and Integration Manual* provides information about the ELA.

GIC CPU interface

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

Vector Processing Unit

The Cortex®-A510 core includes a VPU that is shared between the cores of a dual-core complex. Single-core complexes have a dedicated VPU.

When enabled, the VPU supports Advanced SIMD and floating-point operation. Advanced SIMD is a media and signal processing architecture that adds instructions primarily for audio, video, 3D graphics, and image and speech processing. The floating-point architecture supports single-precision and double-precision floating-point operations. The VPU also supports the *Scalable Vector Extension* (SVE) and SVE2 SIMD instruction sets. SVE and SVE2 complement the Advanced SIMD and floating-point functionality.

Note

The Advanced SIMD architecture, along with its associated implementations and supporting software, are also referred to as Arm® Neon™ technology.

Cryptographic Extension

The Cryptographic Extension is optional in the Cortex®-A510 cores. The Cryptographic Extension adds new instructions to the Advanced SIMD and the 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, SHA-512, and SHA-3
- SM3 hash function and SM4 encryption and decryption
- 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 under a separate license to the Cortex®-A510 core license.

L2 TLB

The L2 TLB is shared between the cores of a dual-core complex, while single-core complexes have a dedicated L2 TLB. The L2 TLB accepts requests from the L1 TLBs and provides *Virtual Address* (VA) to *Physical Address* (PA) translations for instruction side, data side, trace and profiling accesses, and software-accessible address translation operations.

The TLB entries are global or can include *Address Space Identifiers* (ASIDs) to prevent context switch TLB cleans. They also include *Virtual Machine Identifiers* (VMIDs) to prevent TLB cleans on virtual machine switches by the hypervisor. The Cortex®-A510 core can also use the *Common not Private* (CnP) architectural feature that permits cores in a complex to share L2 TLB entries.

L2 memory system

The L2 memory system includes the optional L2 cache. The L2 cache is private to the complex and is 8-way set associative. You can configure the L2 cache size to be 128KB, 192KB, 256KB, 384KB or 512KB. The L2 memory system is connected to the DSU-110 through the CPU bridge.

The L2 cache can be configured to have one or two cache slices. Each slice consists of L2 tag and data RAMs, L2 replacement RAM, L1 duplicate tag RAMs, and associated logic. If two slices are present, most traffic from the cores, the L2 TLB, and from downstream snoops is striped across the slices, based on the value of address bit[6]. This striping increases overall throughput. Accesses to Device non-reorderable memory and to *Distributed Virtual Memory* (DVM) operations are always handled by slice 0.

The data RAMs in each L2 cache slice can be configured to have a single partition or two partitions. Having two partitions increases peak throughput for L2 cache reads and writes by allowing concurrent accesses to different L2 ways.

CPU bridge

In a DynamIQ™ cluster, there is one CPU bridge between each Cortex®-A510 complex and the DSU-110.

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

By default, the CPU bridge is asynchronous to permit different *Power Performance and Area* (PPA) implementation points for each complex. When the CPU bridge runs asynchronously, the core and the DynamIQ™ cluster can run at different frequencies. You can, however, configure the CPU bridge to run synchronously with the memory bus interface without affecting the other asynchronous interfaces such as debug and trace. See *RTL configuration process* in the *Arm® DynamIQ Shared Unit-110 Configuration and Integration Manual* for more information.

Related information

6 Memory management on page 54
7 L1 instruction memory system on page 62
8 L1 data memory system on page 65
9 L2 memory system on page 72
10 Direct access to internal memory on page 76
12 GIC CPU interface on page 85
13 Advanced SIMD and floating-point support on page 87
17 Performance Monitors Extension support on page 99
18 Embedded Trace Extension support on page 116

3.2 Interfaces

The *DynamIQ™ Shared Unit-110* (DSU-110) manages all Cortex®-A510 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®-A510 core implements the Arm®v9.0-A architecture and supports all Arm®v8-A architectures up to Arm®v8.5-A. The Cortex®-A510 core supports the AArch64 Execution state at all Exception levels, EL0 to EL3.

For more information about the programmers model, see:

- *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*
- *Arm®v9-A Supplement for v8-A Arm® Architecture Reference Manual*

Related information

2.4 Supported standards and specifications on page 26

4 Clocks and resets

To provide dynamic power savings, the Cortex®-A510 core supports hierarchical clock gating. It also supports Warm and Cold resets.

Each Cortex®-A510 complex 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. There is one architectural clock gate per core in the complex, and one for the shared logic. If the complex is configured with an asynchronous bridge, the clock input is **COMPLEXCLK<n>**, where **n** indicates the number of the complex within the DSU-110 DynamIQ™ cluster. If the complex is not configured with an asynchronous bridge, the clock input is **SCLK**.

In addition, the Cortex®-A510 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®-A510 core receives the following reset signals from the 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 *Embedded Trace Macrocell* (ETM) logic
    - *Reliability, Availability, and Serviceability* (RAS) logic
- A Cold reset for all logic in the complex, including the debug and *Embedded Trace Macrocell* (ETM) logic.

See *Clocks and resets* and *Power and reset control with Power Policy Units* in the *Arm® DynamIQ Shared Unit-110 Technical Reference Manual* for a complete description of the clock gating and reset scheme of the complex.

5 Power management

The Cortex®-A510 core provides mechanisms to control both dynamic and static power dissipation.

The dynamic power management includes the following features:

- Hierarchical clock gating
- Per-complex *Dynamic Voltage and Frequency Scaling* (DVFS)
- A *Maximum Power Mitigation Mechanism* (MPMM) to control the maximum power

The static power management includes the following features:

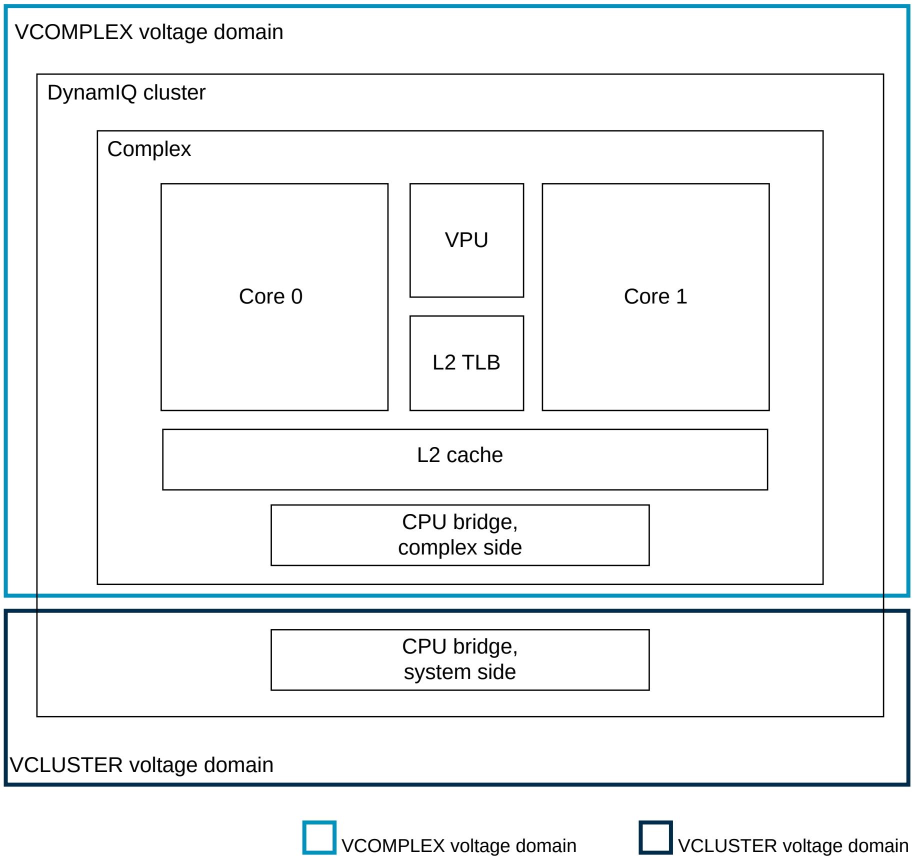
- Powerdown
- Per-complex DVFS
- Dynamic retention, a low-power mode that retains the register and RAM state

5.1 Voltage and power domains

The *DynamIQ™ Shared Unit-110* (DSU-110) *Power Policy Units* (PPUs) control power management for the Cortex®-A510 core. A Cortex®-A510 complex supports separate gated power domains for the complex, for each core inside the complex, and for the *Vector Processing Unit* (VPU). It also supports a dedicated voltage domain for each complex, and a voltage domain for the DSU-110 DynamIQ™ cluster.

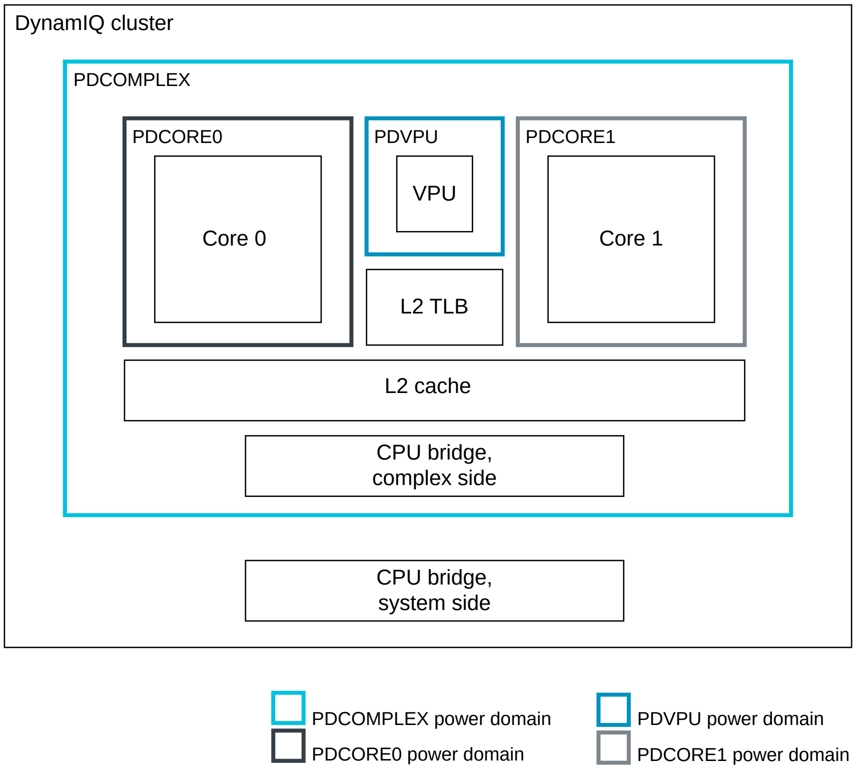
The following figure shows the voltage domains for a Cortex®-A510 configuration with a dual-core complex:

Figure 5-1: Cortex®-A510 voltage domains, dual core



The following figure shows the power domains for an example Cortex®-A510 configuration with a dual-core complex:

Figure 5-2: Cortex®-A510 power domains, dual core



A Cortex®-A510 complex is instantiated within a DynamIQ™ cluster. Within the complex, the system side of the CPU bridge is within the cluster voltage domain, VCLUSTER. From the perspective of the complex, the system side of the CPU bridge is always on. The remainder of the complex logic is in a separate VCOMPLEX voltage domain and PDCOMPLEX power domain. Within the PDCOMPLEX power domain, each core is in the PDCORE<n> power domain, where n is the core instance number. The VPU is in the PDVPU power domain. The rest of the shared logic, consisting of the L2 *Translation Lookaside Buffer* (TLB), the L2 cache, and the CPU bridge, complex side, is in the PDCOMPLEX power domain.

PDCORE<n> is a gated power domain that can support retention. See *The DynamIQ Shared Unit* in the *Arm® DynamIQ Shared Unit-110 Technical Reference Manual* for more information about instance numbering.

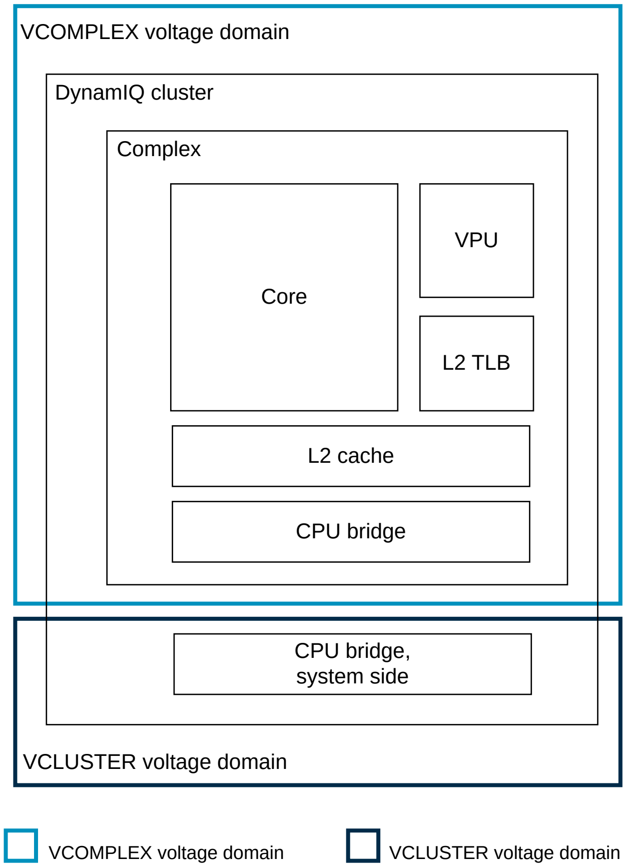
The VCOMPLEX voltage domain operates within a single clock domain, COMPLEXCLK. The CPU bridge contains high-level clock gates and generates gated clocks corresponding to each gated power domain. Also, the clock to the VPU is gated when the VPU is idle.

The CPU bridge can be configured as synchronous or asynchronous. When the CPU bridge is configured as synchronous, the complex runs on **SCLK**, the VCOMPLEX is merged with VCLUSTER, and the complex and the DynamIQ™ cluster are both in the same voltage domain.

The CPU bridge logic within the VCLUSTER voltage domain operates within multiple clock domains. See *Clocks and resets* in the *Arm® DynamIQ Shared Unit-110 Technical Reference Manual* for more information.

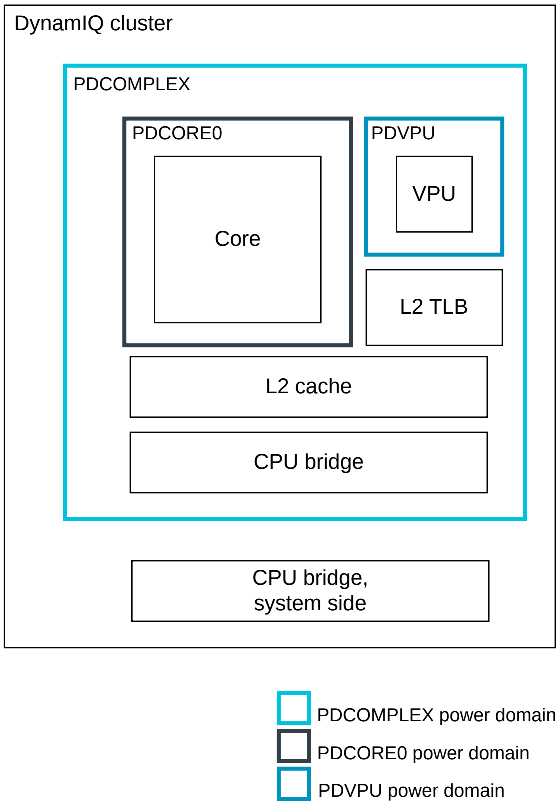
The following figure shows the voltage domains for a Cortex®-A510 configuration with a single-core complex:

Figure 5-3: Cortex®-A510 voltage domains, single core



The following figure shows the power domains for a Cortex®-A510 configuration with a single-core complex:

Figure 5-4: Cortex®-A510 power domains, single core



For a single-core complex, the voltage and power domains are similar to those for a dual-core complex. Within the PDCOMPLEX power domain, the single core is in PDCORE0, a gated power domain that can support retention. The core has its own dedicated logic, including a VPU within its own PDVPU power domain. The L2 TLB, the L2 cache, and the CPU bridge, complex side, is in the PDCOMPLEX power domain.

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 a core within a Cortex®-A510 complex 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` instruction also ensures that store instructions have updated the cache or have been issued to the L3 memory system.

Note

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 Armv8-A architecture profile* for more information about entering low-power state and wakeup events.

5.2.2 Low-power state behavior considerations

You must consider the implications of *Wait for Interrupt* (WFI) and *Wait for Event* (WFE) low-power state behavior that are applicable to a core within a Cortex®-A510 complex.

While the core is in WFI or WFE state, the clocks in the core are temporarily enabled without causing the core to exit WFI or WFE, when any of the following events are detected:

- An access on the Utility bus interface
- A debug access through the APB interface
- A *Generic Interrupt Controller* (GIC) CPU access
- A system snoop request that must be serviced by the core L1 data cache
- Any access from the other core in the complex that must be serviced by the L1 data 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

Each core in a complex can enter WFI or WFE mode separately, leading to the gating of its corresponding core clock. If both cores in the complex are in WFI or WFE mode, the shared logic clock is also gated automatically.

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

5.3 Power control

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

Each core within a Cortex®-A510 complex has an individual PPU for controlling its own core power domain. For example, there is a PPU for PDCORE0 and a PPU for PDCORE1.

In addition, there is a PPU for the DynamIQ™ cluster.

The PPUs decide and request any change in power mode. The targeted core within the Cortex®-A510 complex 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 DynamIQ™ cluster and the cores.

The Cortex®-A510 core includes a *Maximum Power Mitigation Mechanism* (MPMM), which reduces the average power consumed by high-power events in the *Vector Processing Unit* (VPU). Use the Global MPMM Configuration Register to disable MPMM or to change MPMM gears. Use the Global PPM Configuration Register to control whether MPMM control is through the Utility bus or through pin only.

Related information

B.2.1 IMP_CPUPPMCR_EL3, Global PPM Configuration Register on page 185
B.5.44 IMP_CPUMPMMCR_EL3, Global MPMM Configuration Register on page 286
C.1.1 CPUPPMCR, Global PPM Configuration Register on page 379
C.1.2 CPUMPMMCR, Global MPMM Configuration Register on page 380

5.3.1 Maximum Power Mitigation Mechanism

Maximum Power Mitigation Mechanism (MPMM) is a power management feature that detects and limits high activity events, specifically vector unit instructions.

If the count of high-activity events exceeds a pre-defined threshold during an evaluation period, MPMM temporarily limits execution of Advanced SIMD and floating-point instructions.

MPMM provides three gears that enable it to limit certain classes of workloads. Each MPMM gear limits workloads at a different level of aggressiveness, where gear 0 produces the most aggressive throttling and gear 2 the least aggressive. The *Activity Monitoring Unit* (AMU) provides metrics for each gear. An external power controller can use these metrics to budget SoC power in the following ways:

- By limiting the number of cores that can execute higher activity workloads
- By switching to a different *Dynamic Voltage and Frequency Scaling* (DVFS) operating point

MPMM is not intended to limit workloads that operate close to typical power levels. The MPMM event detection and limiting are targeted to limit workloads that operate at significantly higher power levels than typical integer workloads.

Caution

MPMM must not be relied on as the only electrical safety mechanism. It is essentially a localized assistance mechanism that operates at core level. MPMM is not a substitute for a coarse-grained emergency power reduction scheme, but it does minimize the likelihood of such a scheme being engaged. It is a first line of defense rather than a complete solution.

5.4 Core power modes

Each core in a Cortex®-A510 complex, as well as the shared logic, has a defined set of power modes and corresponding legal transitions between these power modes. The power mode of each core can be independent of other cores in a complex or DSU-110 DynamIQ™ cluster.

Power modes for a complex are managed at the DynamIQ™ cluster level as *Power Policy Unit* (PPU) modes. See *Power Management* in the *Arm® DynamIQ Shared Unit-110 Technical Reference Manual* for more information.

The following table shows the supported Cortex®-A510 power modes. It describes the meaning of each mode for a Cortex®-A510 core. Although the power mode can affect any logic that is shared between cores in a Cortex®-A510 complex, the table only describes the effect on the core. See 5.5 Complex power modes on page 50 for more information.

Table 5-1: Cortex®-A510 core power modes

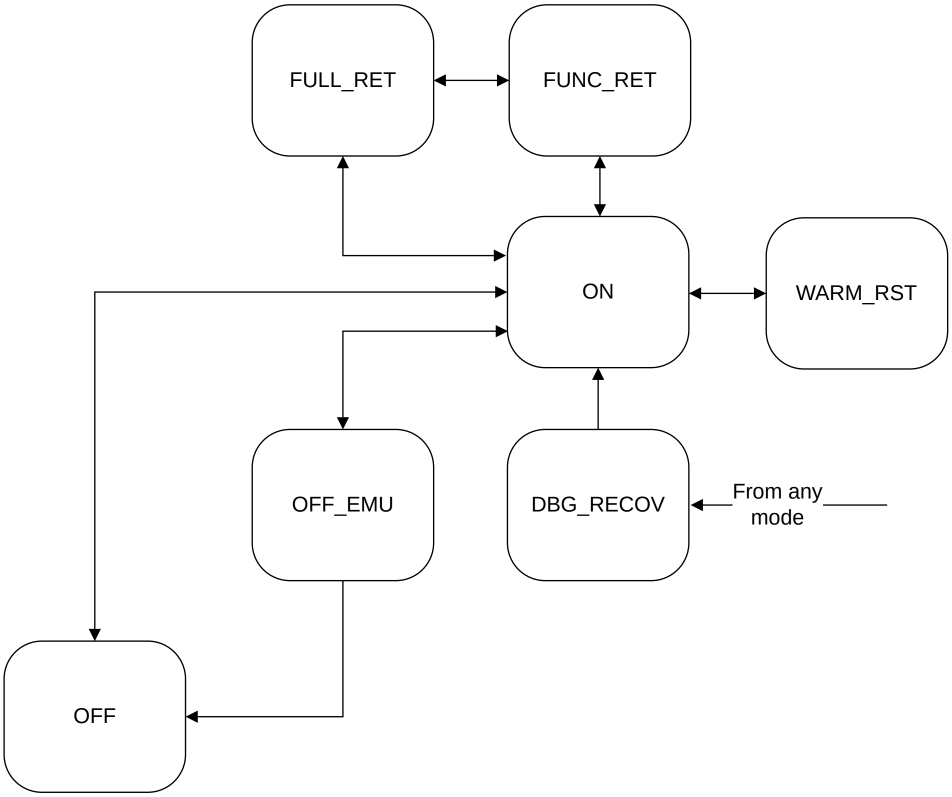
Power mode	Short name	Description
On	ON	The core is powered up and active.
Functional retention	FUNC_RET	The core is fully powered and operational, but the *Vector Processing Unit* (VPU) is idle.
Full retention	FULL_RET	The core is in retention state. In this mode, only power that is required to retain register and RAM state is available. The core is non-operational. A core must be in *Wait for Interrupt* (WFI) or *Wait for Event* (WFE) low-power state before it enters this mode.
Off	OFF	The core is powered down.
Emulated off	OFF_EMU	Emulated off mode permits you to debug the powerup and powerdown cycle without changing the software. In this mode, the core proceeds through all the powerdown steps, except: • The clock is not gated and power is not removed when the core is powered down. • Only the Warm reset is asserted. The debug logic is preserved in the core and remains accessible by the debugger.

Power mode	Short name	Description
Debug recovery	DBG_RECOV	The RAM and logic are powered up. This mode is for applying a Warm reset to the DynamIQ™ cluster, while preserving memory and *Reliability, Availability, and Serviceability* (RAS) registers for debug purposes. Both cache and RAS state are preserved when transitioning from DBG_RECOV to ON. **Caution:** This mode must not be used during normal system operation.
Warm reset	WARM_RST	A Warm reset resets all state except for the debug logic, the *Embedded Trace Macrocell* (ETM) logic, the *Activity Monitor Unit* (AMU) logic, and the RAS registers.

Deviating from the legal power modes can lead to **UNPREDICTABLE** results. You must comply with the dynamic power management and the powerup and powerdown sequences described in 5.6 Cortex-A510 core powerup and powerdown sequence on page 52.

The following figure shows the supported modes for the Cortex®-A510 core and the legal transitions between them:

Figure 5-5: Permitted Cortex®-A510 core power mode transitions



Related information

5.2 Architectural clock gating modes on page 43
5.2.1 Wait for Interrupt and Wait for Event on page 43

5.4.5 Full retention mode on page 49

5.4.1 On mode

In the On power mode, the Cortex®-A510 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. On transition to Off mode, the L1 and L2 caches are disabled, cleaned, and the core is removed from coherency automatically.

Note

If only one core in a complex transitions to Off mode, the L2 cache is not cleaned.

An attempted debug access or Utility bus access when the core domain is off returns an error response on the Utility bus, indicating that the core is not available. See *Utility bus* in the *Arm® DynamIQ Shared Unit-110 Technical Reference Manual* for more information.

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 Functional retention mode

Functional retention mode is a dynamic retention mode that is controlled using IMP_CPUPWRCTLR_EL1. On wakeup, full power to the core can be restored and execution can continue.

In Functional retention mode, the core is fully powered and operational, but the *Vector Processing Unit* (VPU) is in retention state. The VPU can enter this mode if it is idle. Software can enable the Functional retention mode when the retention timer has expired.

If there is a VPU instruction waiting in the execution pipeline, the VPU must exit Functional retention. In a complex where two cores share a VPU, Functional retention only occurs when all cores request it.

Related information

B.1.15 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 161

5.4.5 Full retention mode

Full retention mode is a dynamic retention mode that is controlled using the *Power Policy Units* (PPUs). 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 can enter Full retention mode when all of the following conditions are met:

- The core is in *Wait for Interrupt* (WFI) or *Wait for Event* (WFE) low-power state.
- The retention timer has expired.
- The core clock is not temporarily enabled for L1 or L2 snoops, cache or *Translation Lookaside Buffer* (TLB) maintenance operations, or debug or *Generic Interrupt Controller* (GIC) access.

The core can exit 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 Armv8-A architecture profile*.
- An event that requires the core clock to be temporarily enabled without exiting WFI or WFE low-power state. For example, an L1 or L2 snoop, a cache or TLB maintenance operation, a debug access on the debug APB bus, or a GIC access.

Related information

5.2.1 Wait for Interrupt and Wait for Event on page 43
B.1.15 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 161

5.4.6 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 DSU-110 DynamIQ™ cluster-wide Warm reset must be applied externally. The RAS and cache state are preserved when the core is transitioned to the On mode.

Caution

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

5.4.7 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®-A510 core when the core receives a Warm reset signal from the *DynamIQ™ Shared Unit-110* (DSU-110) side of the CPU bridge.

The Cortex®-A510 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 Armv8-A architecture profile* for more information about RMR_EL3.

5.5 Complex power modes

For a complex containing two cores, a power mode transition in either core requires arbitration between the two cores and their shared logic. The CPU bridge handles this arbitration automatically, without involving the core *Power Policy Unit* (PPU).

The CPU bridge handles system requests for power mode transitions by translating requests into the correct power mode transitions for a particular complex configuration.

The *Power Control State Machine* (PCSM) interface is an external interface for controlling low-level technology-specific power switch and retention controls. See *Power and reset control with Power Policy Units* in the *Arm® DynamIQ Shared Unit-110 Technical Reference Manual* for more information.

The following table shows all possible combinations of core power modes and corresponding power states for a dual-core complex with a shared L2 cache and a *Vector Processing Unit* (VPU).

Table 5-2: PPU mode and power domain states for a dual-core complex

PPU mode		PCSM channel		
Core0	**Core1**	**Core0**	**Core1**	**Shared logic**
On	On	ON	ON	ON
On	Functional retention	ON	ON	ON
On	Full retention	ON	FULL_RET	ON
On	Debug recovery	ON	ON	ON
On	Emulated off	ON	ON	ON
On	Off	ON	OFF	ON
Functional retention	On	ON	ON	ON
Functional retention	Functional retention	ON	ON	FUNC_RET
Functional retention	Full retention	ON	FULL_RET	FUNC_RET
Functional retention	Debug recovery	ON	ON	ON
Functional retention	Emulated off	ON	ON	ON
Functional retention	Off	ON	OFF	FUNC_RET
Full retention	On	FULL_RET	ON	ON
Full retention	Functional retention	FULL_RET	ON	FUNC_RET
Full retention	Full retention	FULL_RET	FULL_RET	FULL_RET
Full retention	Debug recovery	FULL_RET	ON	ON
Full retention	Emulated off	FULL_RET	ON	ON
Full retention	Off	FULL_RET	OFF	FULL_RET
Debug recovery	On	ON	ON	ON
Debug recovery	Functional retention	ON	ON	ON
Debug recovery	Full retention	ON	FULL_RET	ON
Debug recovery	Debug recovery	ON	ON	ON
Debug recovery	Emulated off	ON	ON	ON
Debug recovery	Off	ON	OFF	ON
Emulated off	On	ON	ON	ON
Emulated off	Functional retention	ON	ON	ON
Emulated off	Full retention	ON	FULL_RET	ON
Emulated off	Debug recovery	ON	ON	ON
Emulated off	Emulated off	ON	ON	ON
Emulated off	Off	ON	OFF	ON
Off	On	OFF	ON	ON
Off	Functional retention	OFF	ON	FUNC_RET
Off	Full retention	OFF	FULL_RET	FULL_RET
Off	Debug recovery	OFF	ON	ON
Off	Emulated off	OFF	ON	ON
Off	Off	OFF	OFF	OFF

Note

Emulated off mode operation for a complex is the same as the operation for a core.

In general, any PPU mode combination where only one of the cores is in DBG_RECOV is considered to be a transitional state. In such cases, both cores must eventually go into DBG_RECOV. One exception to this rule is when one core is OFF, in which case it remains OFF while the other core remains in DBG_RECOV.

Deviating from the legal power modes can lead to **UNPREDICTABLE** results. You must comply with the dynamic power management and the powerup and powerdown sequences.

In a dual-core complex, where a single core is being powered down, the shared logic might need to be kept powered on. The powerdown sequence must account for this possibility. Both the L2 cache and the VPU are shared in a dual-core complex. Therefore, when a core in a Cortex®-A510 complex is being powered down:

- If the other core is not Off, the shared logic is kept on and kept in coherency state. Only interfaces that are private to the core are powered down and the core is clock gated.
- If the other core is Off, the powerdown sequence for the complex is the same as the sequence for a single core. This sequence includes taking the complex out of coherency, powering off the shared logic, gating the clocks, and disabling the interfaces.

The following table shows the PCSM power mode and corresponding power modes for the PDCORE0 and PDCORE1 power domains.

Table 5-3: PCSM power states and power modes for core power domains

PCSM power state	PDCORE power mode
ON	On
FULL_RET	Retention
OFF	Off

The following table shows the PCSM power mode and corresponding power modes for the PDCOMPLEX and PDVPU power domains.

Table 5-4: PCSM power states and power modes for complex power domains

PCSM power state	PDCOMPLEX power mode	PDVPU power mode
ON	On	On
FUNC_RET	On	Off
FULL_RET	Retention	Off
OFF	Off	Off

Related information

5.3 Power control on page 44

5.6 Cortex®-A510 core powerup and powerdown sequence

No particular sequence applies to the Cortex®-A510 core powerup. There are no software steps required to bring a core into coherence after reset. For powerdown, the Cortex®-A510 core uses a specific sequence.

To powerdown the Cortex®-A510 core:

1. Save all architectural state.
2. Configure the *Generic Interrupt Controller* (GIC) distributor to disable or reroute interrupts away from the core.
3. Set the IMP_CPUPWRCTLR_EL1.CORE_PWRDN_EN bit to 1 to indicate to the power controller that a powerdown is requested.
4. Execute an `ISB` instruction.
5. Execute a `WFI` instruction.

After executing `WFI` and then receiving a powerdown request from the power controller, the hardware:

- Disables and cleans the L1 cache
- Removes the core from 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 reset is the only way to wake up the core from the *Wait for Interrupt* (WFI) state.

Related information

B.1.15 IMP_CPUPWRCTLR_EL1, CPU Power Control Register on page 161

5.7 Debug over powerdown

The Cortex®-A510 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 *DynamIQ™ Shared Unit-110* (DSU-110). The DebugBlock is external to the DSU-110 DynamIQ™ cluster and must remain powered on during the debug over powerdown process.

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

6 Memory management

The *Memory Management Unit* (MMU) is responsible for translating an input address to an output address. This translation is based on address mapping and memory attribute information that is available in the Cortex®-A510 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®-A510 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®-A510 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.

The Cortex®-A510 core supports the *Common not Private* (CnP) feature. CnP is an architectural feature that permits cores in a complex to share translation tables. When CnP is enabled and in use, all cores in a complex can share L2 TLB entries and make better use of the TLB. Without it, each core in a complex might cache the same translation, reducing the effective size of the TLB.

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

6.1 Memory Management Unit components

The Cortex®-A510 *Memory Management Unit* (MMU) includes several *Translation Lookaside Buffers* (TLBs) and a translation table prefetcher.

A TLB is a cache of recently executed page translations within the MMU. The Cortex®-A510 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	• 16 entries • Fully associative • Located in the L1 instruction memory block • TLB hits return the *Physical Address* (PA) to the instruction cache
L1 data TLB	• 16 entries • Fully associative • Located in the L1 data memory block • TLB hits return the PA to the data cache
L1 *TRace Buffer Extension* (TRBE) TLB	• 2 entries • *Virtual Address* (VA) to PA translations of any table and block size
L2 TLB	• 8-way set associative • A main block that is located within a complex • Shared between the cores of a dual-core complex • Supports dirty bit update, that is, hardware update of access flag and access permissions • Provides translations for instruction side, data side, trace and profiling accesses, and address translation operations
Translation table prefetcher	• Detects access to contiguous translation tables and prefetches the next one • Can be disabled in the IMP_CMPXECTLR_EL1 register

The L2 TLB entries contain a global indicator and an *Address Space Identifier* (ASID) to allow context switches without requiring the TLB to be invalidated. The L2 TLB entries also contain a *Virtual Machine IDentifier* (VMID) to allow virtual machine switches by the hypervisor without requiring the TLB to be invalidated.

Some L2 TLB entries do not have a valid ASID and VMID, because ASID and VMID only apply to the EL1&0 translation regime. Also, ASID does not apply to the *Intermediate Physical Address* (IPA) cache.

To save storage, the L1 TLBs use the context tagging that the L2 TLB provides.

A hit in the L1 instruction TLB provides a single clock cycle access to the translation, and returns the PA to the instruction cache for comparison. If the TLB access does not have the correct access permission, then an Instruction Abort is issued.

A hit in the L1 data TLB provides a single clock cycle access to the translation, and returns the PA to the data cache for comparison. If the TLB access does not have the correct access permission, then a Data Abort is issued.

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.

Related information

B.1.14 IMP_CMPXECTLR_EL1, Complex Extended Control Register on page 157

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 57

6.3 Translation Lookaside Buffer match process

The Arm®v8-A architecture provides support for multiple *Virtual Address* (VA) spaces that are translated differently.

Each *Translation Lookaside Buffer* (TLB) entry is associated with a particular translation regime:

- Secure EL3
- Secure EL2
- Non-secure EL2
- Secure EL2&0
- Non-secure EL2&0
- Secure EL1&0
- Non-secure EL1&0

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

- The entry translation regime matches the current translation regime.
- VA bits[48:N], where N is $\log_2$ of the block size for the translation that is stored in the TLB entry, matches the requested address.

- The *Address Space Identifier* (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 *Virtual Machine Identifier* 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:

- Secure EL2&0
- Non-secure EL2&0
- Secure EL1&0
- Non-secure EL1&0

VMID is relevant for the Secure EL1&0 and Non-secure EL1&0 translation regimes when EL2 is enabled for the corresponding Security state.

A mapping cannot be shared between cores unless the mapping is marked as common. TLB mappings that are marked as common are available only to cores that have *Common not Private* (CnP) enabled:

- A core that has CnP disabled cannot use a TLB mapping that is marked as common.
- A core that has CnP enabled cannot use a TLB mapping that is marked as private.

Note

A core that has CnP enabled is one where the corresponding TTBR<n>_ELx.CnP field for the core is set to 1. For the Secure EL1&0 and Non-secure EL1&0 translation regimes where EL2 is enabled, CnP is enabled when VTTBR_EL2.CnP is set to 1.

6.4 Translation table walks

When the Cortex®-A510 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 translation proceeds by looking up the translation table during a translation table walk.

The following steps describe translation table walks in more detail. When the Cortex®-A510 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, then 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, then the MMU performs a hardware translation table walk.

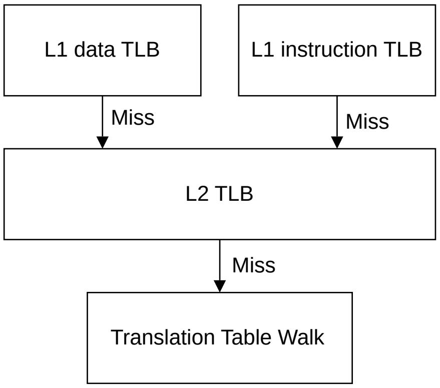
Address translation is performed only when the MMU is enabled. It can also be disabled for a particular translation base register, in which case the MMU returns a translation fault.

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, then 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 62
8 L1 data memory system on page 65
9 L2 memory system on page 72

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®-A510 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®-A510 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. The address captured in the fault address register is that of the address that generated the synchronous external abort.

External aborts are reported asynchronously when they occur during:

- Data accesses that result from load operations to Normal memory
- Load operations to Device memory, including operations that have acquire semantics
- Store operations to any memory type for cache maintenance, TLB invalidate, and instruction cache invalidate operations

Misprogramming contiguous hints

A programmer might program the translation tables incorrectly, 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.

In such cases, the Cortex®-A510 core does not generate a translation fault.

Conflict aborts

The Cortex®-A510 core does not generate Conflict aborts.

6.7 Memory behavior and supported memory types

The Cortex®-A510 core supports memory types defined in the Arm®v8-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 Acknowledgement

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

The following table shows the Device memory types that the Cortex®-A510 core supports.

Table 6-2: Supported Arm®v8-A Device memory types

Memory type	Description
Device-GRE	Device Gathering, Reordering, Early Write Acknowledgement. Device-GRE is similar to Normal Non-cacheable, but does not permit Speculative accesses.
Device-nGRE	Device non-Gathering, Reordering, Early Write Acknowledgement. Transactions might be reordered within the L3 memory system, or in the system interconnect. The use of barriers is required to order accesses to Device-nGRE memory.
Device-nGnRE	Device non-Gathering, non-Reordering, Early Write Acknowledgement. Device-nGnRE is equivalent to the Device memory type in earlier versions of the architecture.
Device-nGnRnE	Device non-Gathering, non-Reordering, No Early Write Acknowledgement. Device-nGnRnE is treated the same as nGnRE inside the Cortex®-A510 core, but reported differently on the bus interface.

Some behaviors are simplified and so for best performance Arm does not recommend using the following memory types:

Write-Through

Memory that is marked as Write-Through is not cached on the data side and does not make coherency requests. On the instruction side, areas that are marked as Write-Through or Write-Back can be cached in the L1 instruction cache.

Mixed Inner and Outer Cacheability

Only memory that is marked as Inner and Outer Write-Back can be cached on the data side and make coherency requests. This rule applies to the memory type only, and not to the allocation hints. All caches within the DSU-110 DynamIQ™ cluster are treated as being part of the Inner Cacheability domain.

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

7 L1 instruction memory system

The Cortex®-A510 core L1 instruction memory system is responsible for fetching instructions and predicting branches. It is part of the *Instruction Fetch Unit* (IFU), which includes a dynamic branch predictor. The L1 instruction memory system includes the L1 instruction cache and the L1 instruction *Translation Lookaside Buffer* (TLB).

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	32KB or 64KB
	4-way set associative
	Virtually-indexed, physically-tagged (VIPT) behaving as *physically-indexed, physically-tagged* (PIPT)
	Single Error Detect (SED) parity cache protection
Cache line length	64 bytes
Cache policy	Pseudo-random cache replacement policy

Note

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 54.

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.

If the instruction cache is disabled, all instruction fetches to cacheable memory are treated as if they were Non-cacheable. This behavior means that instruction fetches might not be coherent with caches in other cores, and software must account for this possibility. Lines might still be allocated into the instruction cache even if the memory is marked as Non-cacheable.

Note

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

5.4.6 Debug recovery mode on page 49

7.2 L1 instruction cache Speculative memory access

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 clean, discarding the currently fetched instructions. On instruction fetches, pages with Device memory type attributes are treated as Non-cacheable Normal memory.

To prevent instruction fetches, Device memory pages must be marked with the translation table descriptor attribute bit *eXecute-Never* (XN). Also, 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 instruction cache is enabled and the instruction fetches miss in the L1 instruction cache, they can still look up in the L1 data cache.

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

7.3 Program flow prediction

The containsCortex®-A510core 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 branch target address.
- For unconditional branches, it only predicts the branch target address.

Program flow prediction hardware contains the following functionality:

- A conditional branch predictor
- An indirect branch predictor
- Dynamic branch predictor history
- The return stack, a stack of nested subroutine return addresses
- A cache that holds the branch target address of previously taken branches

Predicted and non-predicted instructions

Program flow prediction hardware predicts certain branch instructions, including:

- Conditional branches
- Unconditional branches
- Branches that switch between A32 and T32 instruction sets
- Indirect branches that are associated with procedure call and return instructions

The following branch instructions are not predicted:

- Exception return branch instructions
- Data processing instructions that use the *Program Counter* (PC) as a destination register

Return stack

The return stack stores the address and instruction set state. This address is equal to the *Link Register* (LR) value stored in X30.

If predicted, any of the following instructions cause a return stack push:

- `BL`
- `BLR`
- `BLRAA`
- `BLRAAZ`
- `BLRAB`
- `BLRABZ`

Because exception return instructions can change core privilege mode and security state, the following instructions are not predicted:

- `ERET` (exception return)
- `ERETAA`
- `ERETAB`

8 L1 data memory system

The Cortex®-A510 core L1 data memory system is responsible for executing load and store instructions and specific instructions like atomics, cache maintenance operations, and memory tagging instructions. The L1 data memory system includes the L1 data cache and the L1 data *Translation Lookaside Buffer* (TLB).

The L1 data side memory system responds to load and store requests from the *Data Processing Unit* (DPU). It also responds to snoop requests from other cores, or external masters.

The following table shows the L1 data memory system features.

Table 8-1: L1 data memory system features

Feature	Description
Data Cache Unit (DCU)	Manages all load and store operations
	Includes a combined local and global exclusive monitor that is used by Load-Exclusive and Store-Exclusive instructions
STore Buffer (STB)	Handles store instructions and barriers
	Merging store buffer capability which writes to all types of memory, that is, Device, Normal cacheable, and Normal Non-cacheable
Bus Interface Unit (BIU)	Handles the linefills to the L1 data cache
	Receives requests from the cache pipeline in the L1 unit, the STB, and the *Instruction Fetch Unit* (IFU)
	Processes the requests and sends them to the L2 unit
Trace and Profiling Buffer (TPB)	Receives trace data from the *Embedded Trace Macrocell* (ETM) and writes it to memory
Prefetch engine	Detects patterns of cache line requests. Multiple streams are allowed in parallel, capable of detecting both constant requests and patterns of requests.
L1 data cache	32KB or 64KB
	4-way set associative
	Virtually-Indexed, Physically-Tagged (VIPT) behaving as *Physically-Indexed, Physically-Tagged* (PIPT)
	Error Correcting Code (ECC) cache protection
Read path	Dual 128-bit read path from the data L1 memory system to the DPU
Write path	128-bit write path from the DPU to the L1 memory system
Cache line length	64 bytes
Cache policy	Pseudo-random cache replacement policy

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 mode.

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

On a cache miss, the cache performs a critical word-first fill.

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. The `DC CSW` and `DC ISW` instructions perform both a clean and invalidate of the target set/way. The value of HCR_EL2.SWIO has no effect. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information about HCR_EL2.

Note

No relationship between cache sets and physical address can be assumed. Cache maintenance operations by set/way should only be used to invalidate the entire cache

If the data cache is disabled, then:

- Load and store instructions do not access any of the L1 data, L2, or the L3 caches.
- A new line is not allocated in the L2 or L3 caches as a result of an instruction fetch.
- Data cache maintenance operations continue to execute normally.
- Snoop requests continue to access the L1 data, L2, and L3 caches.
- All load and store instructions to cacheable memory are treated as Non-cacheable.

A core cannot disable its L1 and L2 data caches independently. When a core disables the data cache, cacheable memory accesses issued by that core are no longer cached in the L1 or L2 cache. However, another core that shares the L2 cache can still cache data in its L1 cache and in the shared L2 cache.

To maintain data coherency between multiple cores, the Cortex®-A510 core uses the *Modified Exclusive Shared Invalid* (MESI) protocol.

Related information

5.4.6 Debug recovery mode on page 49

8.2 Write streaming mode

The Cortex®-A510 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.

Note

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®-A510 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

B.1.13 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 152

8.3 Memory system implementation

The Cortex®-A510 core supports a single limited order range that includes the entire memory space. It also has specific behavior for transient memory regions.

Instruction implementation in the L1 data memory system

The Cortex®-A510 core supports the atomic instructions added in the Arm®v8.1-A architecture. Atomic instructions to Cacheable memory can be performed either as near atomic or far atomic instructions. Whether a near or far atomic instruction is used depends on the L1 data cache hit and miss information and on the type of operation. Atomic instruction execution location is as follows:

- Near atomic instructions are executed locally, at the L1 memory subsystem level.
- Far atomic instructions are executed in downstream caches and in memory.

Use IMP_CPUECTLR_EL1.ATOM to configure atomic instruction handling. See the *Arm® DynamIQ Shared Unit-110 Technical Reference Manual* for more information about atomic instructions.

The atomic is passed on to the interconnect to perform the operation when all the following conditions apply:

- The interconnect supports far atomics.
- The master interface is configured as CHI.
- The operation misses everywhere within the DSU-110 DynamIQ™ cluster.

If the operation hits anywhere inside the DynamIQ™ cluster, or the interconnect does not support atomics, the L3 memory system performs the atomic operation and allocates the line into the L3 cache if it is not already there.

If there is a requirement to perform a specific atomic operation as a near atomic, you can precede the atomic instruction with a `PRFM PSTL1KEEP` instruction. This brings the line into the cache in a unique state. Using a `PRFM PSTL1KEEP` instruction does not guarantee that the atomic is performed near, as this action is only a performance hint.

The Cortex®-A510 core supports atomics to Device or Non-cacheable memory, however this support relies on the interconnect also supporting atomics. If this type of atomic instruction is executed when the interconnect does not support them, it results in an asynchronous Data Abort.

Transient memory region

The core has a specific behavior for memory regions that are marked as Write-Back cacheable and transient, as defined in the Arm®v8-A architecture.

The transient hint is a qualifier of the cache allocation hints, and indicates that the benefit of caching is for a relatively short period.

For any load that is targeted at a memory region that is marked as transient, the following occurs:

- If the memory access misses in the L1 data cache, the returned cache line is allocated in the L1 data cache but is marked as transient.
- On eviction, if the line is clean and marked as transient, it is not allocated into the L2 cache but is marked as invalid in the L1 data cache.

Use IMP_CPUECTLR_EL1.NTCTL to configure transient and non-temporal L1 eviction.

For stores that are targeted at a memory region that is marked as transient, if the store misses in the L1 data cache, the line is not allocated into the L2 cache.

Non-temporal loads

Non-temporal loads indicate to the caches that the data is likely to be used for only short periods. For example, when streaming single-use read data that is then discarded. In addition to non-temporal loads, there are also prefetch-memory (`PRFM`) hint instructions with the `STRM` qualifier. The Load/Store Non-temporal Pair instructions provide a hint to the memory system that an access is non-temporal or streaming, and unlikely to be repeated in the near future.

Non-temporal loads cause allocation into the L1 data cache, with the same performance as normal loads. However, when a later linefill is allocated into the cache, the cache line that is marked as non-temporal has higher priority to be replaced. To prevent pollution of the L2 cache, a non-temporal line that is evicted from the L1 data cache is not allocated to L2, as would be the case for a normal line. Instead, the non-temporal data is sent directly to the L3 cache. Use IMP_CPUECTLR_EL1.NTCTL to configure transient and non-temporal L1 data cache eviction.

Note

If the core has the line in a unique state, the line is marked as non-temporal in the cache. If the line is shared with other cores, the line is treated normally.

Non-temporal stores are treated the same as stores to a memory region that is marked as transient. That is, if the store misses in the L1 data cache, the line is not allocated into the L2 cache.

Related information

B.1.13 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 152

8.4 Internal exclusive monitor

The Cortex®-A510 core includes an internal exclusive monitor with a 2-state, open and exclusive, state machine that manages Load-Exclusive and Store-Exclusive instructions and Clear-Exclusive instructions.

You can use these instructions to construct semaphores, ensuring synchronization between different processes running on the core. Semaphores can also ensure synchronization 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. The CTR_EL0 register defines the size of the tagged blocks as 16 words, one cache line.

Note

A Load-Exclusive or Store-Exclusive instruction is an instruction that has a mnemonic starting with `LDX`, `LDAX`, `STX`, or `STLX`.

If a Load-Exclusive instruction is performed to Non-cacheable or Device memory, and is to a region of memory in the *System on Chip* (SoC) that does not support exclusive accesses, it causes a Data Abort exception with a Data Fault status code of `0b110101`.

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

Treatment of intervening store operations

When a normal store operation occurs between a Load-Exclusive and a Store-Exclusive instruction from the same core, the normal store does not produce any direct effect on the internal exclusive monitor.

After the Load-Exclusive instruction, the local monitor is in the Exclusive Access state. It remains in the Exclusive Access state after the store. It then returns to the Open Access state only after one of the following operations:

- A Store-Exclusive access
- A `CLREX` instruction
- An exception return

However, if the address that is accessed is in cacheable memory, any eviction of the cache line containing that address clears the monitor. Arm does not recommend placing any load or store instructions between the Load-Exclusive and the Store-Exclusive, because these additional instructions can cause a cache eviction. Any data cache maintenance instruction can also clear the exclusive monitor.

Exclusive monitor

In the exclusive state machine, the transitions are as follows:

- If the monitor is in the Exclusive Access state, and a Store-Exclusive instruction is performed to a different address, then the Store-Exclusive fails and does not update memory.
- If a normal store is performed to a different address, it does not affect the exclusive monitor.
- If a normal store is performed from a different core to the same address, it returns the monitor to the Open Access state. If the store is from the same core, it does not return the monitor to the Open Access state.

Related information

B.5.40 CTR_EL0, Cache Type Register on page 279

8.5 Data prefetching

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

Preload instructions

The Cortex®-A510 core supports the AArch64 Prefetch Memory instruction, `PRFM`.

The `PRFM` and `PLD` instructions signal to the memory system that memory accesses from a specified address are likely to occur soon. The memory system tries to reduce the latency of memory accesses when they occur.

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

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

Hardware data prefetcher

The Cortex®-A510 core has a data prefetch mechanism that looks for cache line fetches with regular or repetitive patterns of data. The core includes multiple data prefetchers. If a data prefetcher detects a pattern, it signals to the memory system that memory accesses from a specified address are likely to occur soon. The memory system responds by starting new linefills to fetch the predicted addresses ahead of the demand loads. These linefills can be in the L1 data cache, the L2 cache, or the L3 cache, depending on which cache the hardware selects.

Prefetch streams end under any of the following circumstances:

- A repetitive pattern is broken.
- A *Data Synchronization Barrier* (DSB) operation is executed.
- A *Wait for Interrupt* (WFI) or *Wait for Event* (WFE) wakeup event is executed.
- A data cache maintenance operation is committed.

The prefetcher is based on virtual addresses. It can therefore cross page boundaries as long as the new page is still cacheable and has read permission.

Data Cache Zero

In the Cortex®-A510 core, the *Data Cache Zero by Virtual Address* (`DC ZVA`) instruction sets a block of 64 bytes in memory, aligned to 64 bytes in size, to `0x00`.

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

9 L2 memory system

The Cortex®-A510 L2 memory system connects the Cortex®-A510 core to the *DynamIQ™ Shared Unit-110* (DSU-110) L3 memory system. It includes an optional unified L2 cache that is private to a complex.

The L2 memory system handles requests from the L1 instruction and data caches, and snoop requests from the L3 memory system. The L2 memory system forwards responses from the L3 system to the core. The core can then take precise or imprecise aborts, depending on the type of transaction.

Note

For some cores, you can implement the DSU-110 to use the Direct connect feature to connect to the core. However, the Cortex®-A510 core does not support Direct connect.

For a complex with two cores, the L2 memory system is shared between the two cores. The L2 memory system also:

- Handles coherent and non-coherent operations from cores and from associated L1 evictions.
- Handles snoop operations from other cores in the DSU-110 DynamIQ™ cluster and from other *Processing Elements* (PEs) in the system, in accordance with the *AMBA® 5 CHI Architecture Specification*.
- Handles instruction cache, *Translation Lookaside Buffer* (TLB), and predictor maintenance operations as *Distributed Virtual Memory* (DVM) messages, including broadcast operations within the complex.

The following table shows the L2 memory system features.

Table 9-1: L2 memory system features

Feature	Type
L2 cache, optional	128KB, 192KB, 256KB, 384KB, or 512KB
	8-way set associative
	Per-complex unified
	Physically-Indexed, *Physically-Tagged* (PIPT)
	Optionally protected with *Error Correcting Code* (ECC)
Cache line length	64 bytes
Cache policy	Dynamic biased cache replacement policy
	Pseudo-exclusive with L1 data caches
	Pseudo-inclusive with L1 instruction caches
Cache protection	Tag, data, and L2 data buffer RAM structures are always protected with ECC.
Cache partitioning	The L2 cache is too small to justify partitioning. The L2 cache stores the *Memory system resource Partitioning And Monitoring* (MPAM) information and propagates it to the L1 and L3 caches.

9.1 Optional integrated L2 cache

You can implement the Cortex®-A510 core with or without an L2 cache.

Allocations into the L2 cache can be made either by the hardware prefetcher, by a `PRFM` instruction, or as a result of stash transactions from the interconnect.

In general, data is allocated to the L2 cache only when evicted from the L1 memory system, not when first fetched from the system. However, there are other cases when data or instructions are allocated to the L2 cache:

- If the Write-Allocate hint is set when the L1 cache enters write-streaming mode, cacheable writes are allocated in the L2 cache until the L2 streaming threshold is reached.
- L2 cache prefetches issued by the L1 caches are allocated in the L2 cache, regardless of the Read-Allocate hint.
- If the Read-Allocate hint is set, cacheable reads from the *Translation Lookaside Buffer* (TLB) or instruction side are allocated in the L2 cache.

Note

This list mentions the most common examples of when data might be allocated to the L2 cache, but it does not include every possible case.

Writes to a memory region that is marked as transient are not allocated to the L2 cache.

When non-temporal data is evicted from the L1 memory system, the data is sent directly to the L3 cache and is not allocated in the L2 cache. Use IMP_CPUECTLR_EL1.NTCTL to configure transient and non-temporal L1 eviction.

L2 cache RAMs are invalidated automatically at reset unless the Debug recovery mode is used.

Related information

5.4.6 Debug recovery mode on page 49
8.2 Write streaming mode on page 66
B.1.13 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 152

9.2 Support for memory types

The Cortex®-A510 core simplifies the 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. All other memory types are not cached.

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

An allocating read to the L1 data cache that has the transient bit set is allocated in the L1 cache. These types of read are marked as most likely to be evicted, according to the L1 eviction policy.

Writes that have the transient bit set are not allocated to the L1 cache or to the L2 cache.

Evictions from L1 cache that is marked as transient are not allocated to the L2 cache.

Use IMP_CPUECTLR_EL1.NTCTL to configure transient and non-temporal L1 eviction.

The standard CHI attributes are passed to the *DynamIQ™ Shared Unit-110* (DSU-110) with no modifications, except for translating the following architectural attributes to CHI attributes:

- Allocate hint
- Shareability
- Cacheability

Note

Inner and Outer cacheability is merged together, as the Cortex®-A510 core only allocates memory that is marked as both Inner and Outer cacheable.

Related information

B.1.13 IMP_CPUECTLR_EL1, CPU Extended Control Register on page 152

9.3 Transaction capabilities

The interface between the Cortex®-A510 L2 memory system and the *DynamIQ™ Shared Unit-110* (DSU-110) defines the transaction capabilities for the core.

The following table shows the maximum values for read, write, *Distributed Virtual Memory* (DVM) issuing, and snoop capabilities of the Cortex®-A510 L2 cache. The table includes values for single-slice L2 cache and dual slice L2 cache configurations.

Table 9-2: Cortex®-A510 L2 cache transaction capabilities

Attribute	Maximum value	Description
Write issuing capability	40, for single slice 80, for dual slice	Maximum number of outstanding write transactions. **Note:** This value depends on the counting method that is used, but typical values are quoted.
Read issuing capability	31, for single slice 48, for dual slice	Maximum number of outstanding read transactions.

Attribute	Maximum value	Description
Snoop acceptance capability	29, for single slice 49, for dual slice	Maximum number of outstanding snoops accepted.
DVM issuing capability	18, for single slice 36, for dual slice	Maximum number of outstanding DVM operation transactions.

10 Direct access to internal memory

The Cortex®-A510 core provides a mechanism to read the internal memory that the L1 and L2 cache and *Translation Lookaside Buffer* (TLB) structures use, through **IMPLEMENTATION DEFINED** System registers. When the coherency between the cache data and the system memory data is broken, you can use this mechanism to investigate any issues.

Direct access to internal memory is available only in EL3. In all other modes, accessing these registers results in an Undefined Instruction exception. Use the **IMPLEMENTATION DEFINED** system registers to select the appropriate memory block and location. The following table shows the System register operations that read the data and the information that the cache data includes.

Table 10-1: IMPLEMENTATION DEFINED System registers for accessing internal memory

Name	Access encoding	Operation	Read data content
IMP_CDBGDR0_EL3	`MRS <Xt>, S3_6_C15_C0_0`	Store data from a preceding cache debug operation	Data
SYS IMP_CDBGL1DCTR	`SYS #6, C15, C2, #0, <Xt>`	Read contents of L1 data cache tag RAM	Data
SYS IMP_CDBGL1ICTR	`SYS #6, C15, C2, #1, <Xt>`	Read contents of L1 instruction cache tag RAM	Set and way
SYS IMP_CDBGL2TR0	`SYS #6, C15, C2, #2, <Xt>`	Read contents of L2 TLB	Index and way
SYS IMP_CDBGL2CTR	`SYS #6, C15, C2, #3, <Xt>`	Read contents of L2 cache tag RAM	Index and way
SYS IMP_CDBGL1DCDTR	`SYS #6, C15, C2, #4, <Xt>`	Read contents of L1 data cache dirty RAM	Set and way
SYS IMP_CDBGL1DCMR	`SYS #6, C15, C3, #0, <Xt>`	Read contents of L1 data cache *Memory Tagging Extension* (MTE) tag RAM	Set and way
SYS IMP_CDBGL2TR1	`SYS #6, C15, C3, #2, <Xt>`	Read contents of L2 TLB	Index and way
SYS IMP_CDBGL2CMR	`SYS #6, C15, C3, #3, <Xt>`	Read contents of L2 cache MTE tag RAM	Set and way
SYS IMP_CDBGL1DCDR	`SYS #6, C15, C4, #0, <Xt>`	Read contents of L1 data cache data RAM	Set, way, and offset
SYS IMP_CDBGL1ICDR	`SYS #6, C15, C4, #1, <Xt>`	Read contents of L1 instruction cache data RAM	Set, way, and offset
SYS IMP_CDBGL2TR2	`SYS #6, C15, C4, #2, <Xt>`	Read contents of L2 TLB	Index and way
SYS IMP_CDBGL2CDR	`SYS #6, C15, C4, #3, <Xt>`	Read contents of L2 cache data RAM	Set, way, and offset

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 for locating the cache data entry for tag and data memory is set in `Xn` in the appropriate `SYS` instruction.

To read the data from a particular RAM, write to the appropriate System register using the encoding shown in the table in 10 Direct access to internal memory on page 76.

For example, to read the data from the L1 data cache tag RAM, access IMP_CDBGL1DCTR as follows:

```
SYS #6, C15, C2, #0, <Xt>
```

To specify the cache line from which you want to read, use the bit description table in the System register description.

The cache tag specified is written to IMP_CDBGDR0_EL3.

Related information

B.3.1 IMP_CDBGDR0_EL3, Cache Debug Data Register 0 on page 187
B.4.1 SYS IMP_CDBGL1DCTR, L1 Data Cache Tag Read Operation on page 198
B.4.2 SYS IMP_CDBGL1ICTR, L1 Instruction Cache Tag Read Operation on page 199
B.4.5 SYS IMP_CDBGL1DCDTR, L1 Data Cache Dirty Read Operation on page 203
B.4.6 SYS IMP_CDBGL1DCMR, L1 Data Cache MTE Tag Read Operation on page 204
B.4.9 SYS IMP_CDBGL1DCDR, L1 Data Cache Data Read Operation on page 208
B.4.10 SYS IMP_CDBGL1ICDR, L1 Instruction Cache Data Read Operation on page 210

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 `Xn` in the appropriate `SYS` instruction.

To read the data from a particular RAM, write to the appropriate System register using the encoding shown in the table in 10 Direct access to internal memory on page 76.

For example, to read the data from the L2 cache tag RAM, access IMP_CDBGL2CTR as follows:

```
SYS #6, C15, C2, #3, <Xt>
```

To specify the cache line from which you want to read, use the bit description table in the System register description.

The cache tag specified is written to IMP_CDBGDR0_EL3.

Related information

B.3.1 IMP_CDBGDR0_EL3, Cache Debug Data Register 0 on page 187
B.4.4 SYS IMP_CDBGL2CTR, L2 Cache Tag Read Operation on page 202
B.4.8 SYS IMP_CDBGL2CMR, L2 Cache MTE Tag Read Operation on page 207
B.4.12 SYS IMP_CDBGL2CDR, L2 Cache Data Read Operation on page 212

10.3 L2 TLB encodings

The L2 *Translation Lookaside Buffer* (TLB) is 8-way set associative and is RAM-based. Individual TLB entries can be read into the data registers by executing the IMP_CDBGL2TDR operation.

The encoding that is used to locate the data for a TLB is set in `Xn` in the appropriate `SYS` instruction.

To read the data from a particular TLB, write to the appropriate System register using the encoding shown in the table in 10 Direct access to internal memory on page 76.

For example, to read bits[63:0] from the L2 TLB, access IMP_CDBGL2TR0 as follows:

```
SYS #6, C15, C2, #2, <Xt>
```

To specify the cache line from which you want to read, use the bit description table in the System register description.

The cache tag specified is written to IMP_CDBGDR0_EL3.

Related information

B.3.1 IMP_CDBGDR0_EL3, Cache Debug Data Register 0 on page 187
B.4.3 SYS IMP_CDBGL2TR0, L2 TLB Read Operation 0 on page 201
B.4.7 SYS IMP_CDBGL2TR1, L2 TLB Read Operation 1 on page 206
B.4.11 SYS IMP_CDBGL2TR2, L2 TLB Read Operation 2 on page 211

11 RAS extension support

The Cortex®-A510 core implements the *Reliability, Availability, Serviceability* (RAS) extension, including all extensions up to Arm®v9.0-A.

The Cortex®-A510 core supports the following RAS extension features:

- *Fault Handling Interrupts* (FHIs)
- *Error Recovery Interrupts* (ERIs)
- Poison attribute on bus transfers
- Cache protection with *Single Error Detect* (SED) parity
- Cache protection with *Single Error Correct Double Error Detect* (SECDED) *Error Correcting Code* (ECC)
- Error record registers to help software perform recovery actions
- Error injection capabilities to facilitate software and system debug
- The *Error Synchronization Barrier* (ESB) instruction to synchronize unrecoverable errors

Each of the Cortex®-A510 core RAMs has either cache protection with SECDED ECC or cache protection with SED parity, as defined in 11.1 Cache protection behavior on page 79.

After an `ESB` instruction, the core ensures that all SError interrupts that are generated by instructions before the `ESB` are either taken or deferred. If the core cannot take the interrupt, it records it in the Deferred Interrupt Status Register DISR_EL1. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information on DISR_EL1.

Fault detection features are included in groups within the DSU-110 DynamIQ™ cluster and the Cortex®-A510 core. Each group of fault detection features is referred to as a node. The following nodes are implemented in the Cortex®-A510 core and the DynamIQ™ cluster:

- Node 0 contains fault detection features that are located within the shared L3 memory system in the *DynamIQ™ Shared Unit-110* (DSU-110)
- Node 1 contains fault detection features that are located within the private L1 memory systems in the Cortex®-A510 core
- Node 2 contains fault detection features that are located within the shared L2 memory systems in the complex

The Cortex®-A510 core RAS registers correspond to either node 1 or node 2, as indicated by the register name.

For more information on the architectural RAS Extension and nodes, 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, Serviceability* (RAS) Extension that is implemented in the Cortex®-A510 core includes cache protection. In this case, the Cortex®-A510 core protects against errors that result in a RAM bitcell holding the incorrect value.

The RAMs in the Cortex®-A510 core have the following capabilities:

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 Error Correcting Code

The following table shows which protection type is applied to each RAM in the Cortex®-A510 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
L1 data cache data	SECDED ECC
L1 data cache tag	SED Parity
Duplicate L1 data cache tag	SECDED ECC
L1 data cache dirty	SECDED ECC
L2 *Translation Lookaside Buffer* (TLB)	SED Parity
L2 cache data	SECDED ECC
L2 cache tag	SECDED ECC
L2 data buffer	SECDED ECC

If there are multiple single-bit errors in different RAMs, or within different protection granules within the same RAM, then the core 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 and either reports or defers 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, possibly causing data corruption.

If there are three or more bit errors within the same protection granule, the core might or might not detect the errors. Whether it detects the errors or not depends on the RAM and the position of the errors within the RAM. The cache protection feature of the core has a minimal performance impact when no errors are present.

11.2 Error containment

The Cortex®-A510 core supports error containment, which means that an error is detected and not silently propagated.

Error containment also provides 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®-A510 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 ERR*n*CTLR.FI is set, all detected Deferred errors, Uncorrected errors, and overflows of the corrected error counters generate an FHI. When ERR*n*CTLR.CFI is set, all detected Corrected errors also generate an FHI.

FHIs from core *n* are signaled using **nCOREFAULTIRQ[n]**, and FHIs from complex *n* are signaled using **nCOMPLEXFAULTIRQ[n]**.

Error recovery interrupts

When ERR*n*CTLR.UI is set, all detected Uncorrected errors that are not deferred generate an ERI.

ERIs from core *n* are signaled using **nCOREERRIRQ[n]**, and ERIs from complex *n* are signaled using **nCOMPLEXERRIRQ[n]**.

Related information

11.7 RAS registers 2 on page 83
C.5.1 ERR2CTLR, Error Record Control Register on page 465
C.5.2 ERR1CTLR, Error Record Control Register on page 468
C.5.14 ERR1STATUS, Error Record Primary Status Register on page 499

11.4 Error detection and reporting

When the Cortex®-A510 core consumes an error, it raises different exceptions depending on the error type.

The Cortex®-A510 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 Control Registers, ERR1CTLR and ERR2CTLR

These registers enable error reporting and also enable various interrupts that are related to errors and faults.

Error Record Feature Registers, ERR1FR and ERR2FR

These read-only registers specify various error record settings.

Error Record Miscellaneous Registers 0-3, ERR*n*MISC0-3

These **IMPLEMENTATION DEFINED** error syndrome registers might record details of the error location and counts.

Pseudo-fault Generation Feature register, ERR1PFGF and ERR2PFGF

These registers define which common architecturally defined fault generation features are implemented.

Pseudo-fault Generation Control register, ERR1PFGCTL and ERR2PFGCTL

These registers enable controlled fault generation.

Error Record Primary Status Registers, ERR1STATUS and ERR2STATUS

These registers contain status information for the error record.

See 11.7 RAS registers 2 on page 83 for a complete list of these registers.

Error reporting and performance monitoring

All detected memory errors and *Error Correcting Code* (ECC) or parity errors trigger the MEMORY_ERROR event.

The *Performance Monitoring Unit* (PMU) counters count the MEMORY_ERROR event, provided the event is selected and the counter is enabled. In Secure state, the event is counted only if MDCR_EL3.SPME is asserted.

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®-A510 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 dirty 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, ERR0PFGCDN. 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 ERR0PFGCDN.

Note

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

11.6 RAS registers 1

The summary table provides an overview of *Reliability, Availability, and Serviceability* (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

11.7 RAS registers 2

The summary table provides an overview of Reliability, Availability, and Serviceability (RAS) registers in the core. Individual register descriptions provide detailed information.

Table 11-3: RAS register summary

Name	Reset	Width	Description
ERR2CTLR	See individual bit resets.	64-bit	Error Record Control Register
ERR1CTLR	See individual bit resets.	64-bit	Error Record Control Register
ERR2FR	See individual bit resets.	64-bit	Error Record Feature Register
ERR1FR	See individual bit resets.	64-bit	Error Record Feature Register
ERR2MISC1	0x0	64-bit	Error Record Miscellaneous Register 1
ERR1MISC1	0x0	64-bit	Error Record Miscellaneous Register 1
ERR2MISC0	See individual bit resets.	64-bit	Error Record Miscellaneous Register 0
ERR1MISC0	See individual bit resets.	64-bit	Error Record Miscellaneous Register 0
ERR2PFGF	See individual bit resets.	64-bit	Pseudo-fault Generation Feature Register
ERR1PFGF	See individual bit resets.	64-bit	Pseudo-fault Generation Feature Register
ERR2PFGCTL	See individual bit resets.	64-bit	Pseudo-fault Generation Control Register
ERR1PFGCTL	See individual bit resets.	64-bit	Pseudo-fault Generation Control Register
ERR2STATUS	See individual bit resets.	64-bit	Error Record Primary Status Register
ERR1STATUS	See individual bit resets.	64-bit	Error Record Primary Status Register
ERR2MISC3	0x0	64-bit	Error Record Miscellaneous Register 3
ERR1MISC3	0x0	64-bit	Error Record Miscellaneous Register 3
ERR2MISC2	0x0	64-bit	Error Record Miscellaneous Register 2
ERR1MISC2	0x0	64-bit	Error Record Miscellaneous Register 2

12 GIC CPU interface

The *Generic Interrupt Controller* (GIC) supports and controls interrupts. The GIC distributor connects to the Cortex®-A510 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 DSU-110 DynamIQ™ cluster has a GIC CPU interface, which connects to a common external distributor component.

The GICv4.1 architecture implemented in the Cortex®-A510 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®-A510 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 GIC architectures other than the GICv4.1 architecture. If the Cortex®-A510 core is not integrated with an external GICv4.1 interrupt distributor component 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 GIC register summary

The summary table provides an overview of *Generic Interrupt Controller* (GIC) registers in the Cortex®-A510 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®-A510 core supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set without floating-point exception trapping.

The Cortex®-A510 core floating-point implementation includes Armv8-A architecture features, as specified in 2.4 Supported standards and specifications on page 26.

14 Scalable Vector Extensions support

The Cortex®-A510 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®-A510 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®v9-A Supplement for v8-A Arm® Architecture Reference Manual* 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 also accessible through the external debug interface or Utility bus interface.

15.1 Generic system control register summary

The summary table provides an overview of generic system control registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table 15-1: 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)
IMP_CPUACTLR_EL1	3	C15	0	C1	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register
IMP_CPUACTLR2_EL1	3	C15	0	C1	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 2
IMP_CMPXACTLR_EL1	3	C15	0	C1	3	See individual bit resets.	64-bit	Complex Auxiliary Control Register

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
IMP_CPUECTLR_EL1	3	C15	0	C1	4	See individual bit resets.	64-bit	CPU Extended Control Register
IMP_CMPXECTLR_EL1	3	C15	0	C1	7	See individual bit resets.	64-bit	Complex Extended Control Register
IMP_CPUPWRCTLR_EL1	3	C15	0	C2	7	See individual bit resets.	64-bit	CPU Power Control Register
IMP_ATCR_EL2	3	C15	4	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register
IMP_AVTCR_EL2	3	C15	4	C7	1	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register
IMP_ATCR_EL1	3	C15	5	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register
IMP_ATCR_EL3	3	C15	6	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation 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)

16 Debug

The DSU-110 DynamIQ™ 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.

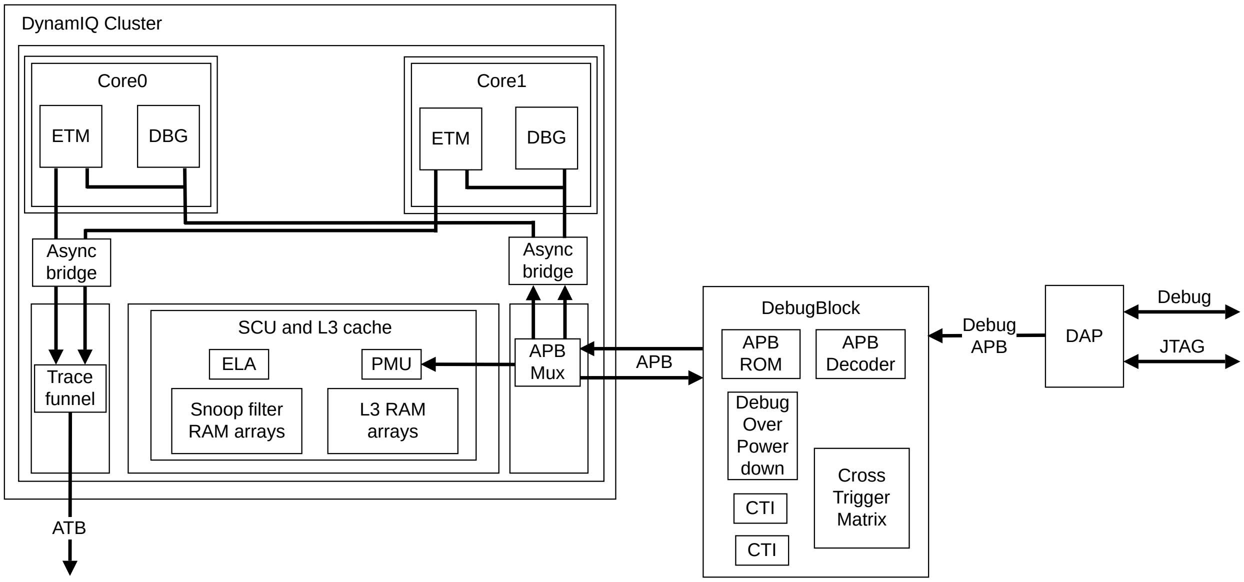
The connection between the cluster and the DebugBlock consists of a pair of *Advanced Peripheral Bus* APB interfaces, one in each direction. All debug traffic, except the authentication interface, takes place over this interface as read or write APB transactions. This debug traffic includes register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

The debug system implements the following CoreSight debug components:

- Per-core *Embedded Trace Macrocell* (ETM), 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 ETM 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.

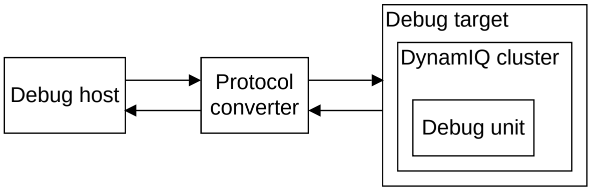
The Cortex®-A510 core also supports direct access to internal memory, that is, cache debug. Direct access to internal memory allows software to read the internal memory that the L1 and L2 cache and *Translation Lookaside Buffer* (TLB) structures use. See 10 Direct access to internal memory on page 76 for more information.

16.1 Supported debug methods

The DSU-110 DynamIQ™ cluster along with its associated complexes and 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 A510 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 A510 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®-A510 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

5.7 Debug over powerdown on page 53

16.2.1 Core interfaces

Except for the Trace registers, all the Debug register groups are both System register based and memory-mapped. System register access allows the Cortex®-A510 core to access certain Debug registers directly.

Access to the Debug registers is partitioned as follows:

Debug

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

You can access the performance monitor registers using the APB slave port that connects into the DebugBlock of the DSU.

Activity monitoring

You can access the activity monitor registers using the Utility bus interface.

Trace

You can access the *Embedded Trace Macrocell* (ETM) registers using the APB slave port that connects into the DebugBlock of the DSU.

ELA registers

You can access the ELA registers using the APB slave port that connects into the DebugBlock of the DSU.

Note

This function is memory-mapped and is not accessible using System registers.

See *Interfaces* and *Debug* in the *Arm® DynamIQ Shared Unit-110 Technical Reference Manual* for information on the APB slave port interface and the Utility bus interface.

Related information

17.4 Performance monitors register summary on page 114
17.5 Memory-mapped PMU register summary on page 114
18.8 ETE register summary on page 122
20.5 Memory-mapped AMU register summary on page 127
20.4 Activity monitors register summary on page 127
16.8 Debug register summary on page 98

16.2.2 Effects of resets on Debug registers

Cold and Warm resets are generated within the DSU-110 DynamIQ™ cluster and have different effects on the Debug registers.

A Cold reset includes reset of the core logic and the integrated debug functionality. It initializes the core logic, including the *Embedded Trace Macrocell* (ETM) trace unit and debug logic.

A Warm reset includes reset of the core logic but not the debug, ETM, or *Activity Monitoring Unit* (AMU) logic, or the *Reliability, Availability, and Serviceability* (RAS) registers.

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 *Read-As-One* (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.
Default	-	This is normal access, none of the conditions apply.

16.2.4 Breakpoints and watchpoints

The Cortex®-A510 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®-A510 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®-A510 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 DSU-110 DynamIQ™ cluster level.

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

16.5 ROM table

The Cortex®-A510 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. There is one ROM table for each complex 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 DynamIQ™ cluster and one for the DebugBlock, and has entry points in the cluster ROM table for the ROM tables belonging to each core or complex. See *ROM tables* in the *Arm® DynamIQ Shared Unit-110 Technical Reference Manual* for more information.

The Cortex®-A510 core ROM table includes the following entries:

Table 16-2: Core ROM table

Offset	Name	Description
`0x0000`	ROMENTRY0	Core 0 debug
`0x0004`	ROMENTRY1	Core 0 *Performance Monitoring Unit* (PMU)
`0x0008`	ROMENTRY2	Core 0 *Embedded Trace Macrocell* ETM
`0x000C`	ROMENTRY3	Optional *Embedded Logic Analyzer* (ELA)
`0x0010`	ROMENTRY4	Core 1 debug
`0x0014`	ROMENTRY5	Core 1 PMU
`0x0018`	ROMENTRY6	Core 1 ETM
`0x001C`	ROMENTRY7	-

Related information

16.7 ROM table register summary on page 97

16.6 CoreSight component identification

Each component associated with the Cortex®-A510 core has a unique set of CoreSight ID values.

Table 16-3: Cortex®-A510 CoreSight component identification

Component	Peripheral ID	Component ID	DevType	DevArch	Revision
ETM	0x04300BBD46	0xB105900D	0x13	0x47705A13	r0p3
PMU	0x04300BBD46	0xB105900D	0x16	0x47702A16	r0p3
DBG	0x04300BBD46	0xB105900D	0x15	0x47709A15	r0p3
CTI	0x04003BBD46	0xB105900D	0x14	0x47711A14	r3p0
ROM table	0x04300BBD46	0xB105900D	0x00	0x47700AF7	r0p3

Note

The CTI revision and peripheral ID values depend on the revision of the DSU. The values in the table above are for the r3p0 revision of the DSU-110.

16.7 ROM table register summary

The summary table provides an overview of memory-mapped ROM table registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table 16-4: ROM table register summary

Offset	Name	Reset	Width	Description
0x0	ROMENTRY0	See individual bit resets	32-bit	Class 0x9 ROM table entries
0x4	ROMENTRY1	See individual bit resets	32-bit	Class 0x9 ROM table entries
0x8	ROMENTRY2	See individual bit resets	32-bit	Class 0x9 ROM table entries
0xC	ROMENTRY3	See individual bit resets	32-bit	Class 0x9 ROM table entries
0x10	ROMENTRY4	See individual bit resets	32-bit	Class 0x9 ROM table entries
0x14	ROMENTRY5	See individual bit resets	32-bit	Class 0x9 ROM table entries
0x18	ROMENTRY6	See individual bit resets	32-bit	Class 0x9 ROM table entries
0x1C	ROMENTRY7	See individual bit resets	32-bit	Class 0x9 ROM table entries
0xFBC	DEVARCH	See individual bit resets	32-bit	Device Architecture Register
0xFC0	DEVID2	0x0	32-bit	Device Configuration Register 2
0xFC4	DEVID1	0x0	32-bit	Device Configuration Register 1
0xFC8	DEVID	See individual bit resets	32-bit	Device Configuration Register
0xFCC	DEVTYPE	See individual bit resets	32-bit	Device Type Register
0xFD0	PIDR4	See individual bit resets	32-bit	Peripheral Identification Register 4
0xFD4	PIDR5	0x0	32-bit	Peripheral Identification Register 5
0xFD8	PIDR6	0x0	32-bit	Peripheral Identification Register 6
0xFDC	PIDR7	0x0	32-bit	Peripheral Identification Register 7

Offset	Name	Reset	Width	Description
0xFE0	PIDR0	See individual bit resets	32-bit	Peripheral Identification Register 0
0xFE4	PIDR1	See individual bit resets	32-bit	Peripheral Identification Register 1
0xFE8	PIDR2	See individual bit resets	32-bit	Peripheral Identification Register 2
0xFEC	PIDR3	See individual bit resets	32-bit	Peripheral Identification Register 3
0xFF0	CIDR0	See individual bit resets	32-bit	Component Identification Register 0
0xFF4	CIDR1	See individual bit resets	32-bit	Component Identification Register 1
0xFF8	CIDR2	See individual bit resets	32-bit	Component Identification Register 2
0xFFC	CIDR3	See individual bit resets	32-bit	Component Identification Register 3

Related information

16.5 ROM table on page 96

16.8 Debug register summary

The summary table provides an overview of memory-mapped debug registers in the Cortex®-A510 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

17 Performance Monitors Extension support

The Cortex®-A510 core implements the Performance Monitors Extension, including Arm®v8.4-A and Arm®v8.5-A performance monitoring features.

The Cortex®-A510 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 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®-A510 core *Performance Monitoring Unit* (PMU) collects events from other units in the design and uses numbers to reference these events.

17.1.1 Architectural performance monitors events

The Cortex®-A510 core *Performance Monitoring Unit* (PMU) collects architecturally defined events.

The following table lists the Cortex®-A510 architectural performance monitors events. See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information about these events.

See also the *Arm®v9-A Supplement for v8-A Arm® Architecture Reference Manual* for information about SVE-specific PMU events.

Note

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

Table 17-1: Architectural PMU events

Event number	Event mnemonic	Description
`0x0000`	SW_INCR	Software increment. This event counts any instruction architecturally executed (condition code check pass).
`0x0001`	L1I_CACHE_REFILL	L1 instruction cache refill. This event counts any instruction fetch that misses in the cache. The following instructions are not counted: • Cache maintenance instructions • Non-cacheable accesses
`0x0002`	L1I_TLB_REFILL	L1 instruction TLB refill. This event counts any refill of the instruction L1 TLB from the L2 TLB, including refills that result in a translation fault. TLB maintenance instructions are not counted. This event counts regardless of whether the *Memory Management Unit* (MMU) is enabled.
`0x0003`	L1D_CACHE_REFILL	L1 data cache refill. This event counts any load or store operation or translation table walk that causes data to be read from outside the L1 cache. The event includes accesses that do not allocate into the L1 cache. The following instructions are not counted: • Cache maintenance instructions and prefetches • Stores of an entire cache line, even if they make a coherency request outside the L1 cache • Partial cache line writes that do not allocate into the L1 cache • Non-cacheable accesses This event counts the sum of L1D_CACHE_REFILL_RD and L1D_CACHE_REFILL_WR.
`0x0004`	L1D_CACHE	L1 data cache access. This event counts any load or store operation or translation table walk that looks up in the L1 data cache. In particular, any access that could count the L1D_CACHE_REFILL event causes this event to count. The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses This event counts the sum of L1D_CACHE_RD and L1D_CACHE_WR.

Event number	Event mnemonic	Description
0x0005	L1D_TLB_REFILL	L1 data TLB refill. This event counts any refill of the L1 data TLB from the L2 TLB, including refills that result in a translation fault. TLB maintenance instructions are not counted. This event counts regardless of whether the MMU is enabled.
0x0006	LD_RETIRED	Instruction architecturally executed, condition code check pass, load. This event counts all load and prefetch instructions, including the Arm®v8.1-A atomic instructions, other than the ST* variants.
0x0007	ST_RETIRED	Instruction architecturally executed, condition code check pass, store. This event counts all store instructions and the Data Cache Zero by Virtual Address (DC ZVA) instruction. The event includes all the Arm®v8.1-A atomic instructions. Store-Exclusive instructions that fail are not counted.
0x0008	INST_RETIRED	Instruction architecturally executed. This event counts all retired instructions, including ones that fail their condition check.
0x0009	EXC_TAKEN	Exception taken.
0x000A	EXC_RETURN	Instruction architecturally executed, condition code check pass, exception return.
0x000B	CID_WRITE_RETIRED	Instruction architecturally executed, condition code check pass, write to CONTEXTIDR. This event only counts writes using the CONTEXTIDR_EL1 mnemonic. Writes to CONTEXTIDR_EL12 are not counted.
0x000C	PC_WRITE_RETIRED	Instruction architecturally executed, condition code check pass, software change of the Program Counter. This event counts all taken branches, excluding exception entries or breakpoint instructions.
0x000D	BR_IMMED_RETIRED	Instruction architecturally executed, immediate branch. This event counts all branches decoded as immediate branches, taken or not, excluding exception entries and debug entries.
0x000E	BR_RETURN_RETIRED	Instruction architecturally executed, condition code check pass, procedure return.
0x0010	BR_MIS_PRED	Mispredicted or not predicted branch speculatively executed. This event counts any predictable branch instruction that is mispredicted for either of the following reasons: • Dynamic misprediction • The MMU is off and the branches are statically predicted not taken
0x0011	CPU_CYCLES	Cycle. This event is not exported to the ETM.
0x0012	BR_PRED	Predictable branch speculatively executed. This event counts all predictable branches.

Event number	Event mnemonic	Description
0x0013	MEM_ACCESS	Data memory access. This event counts memory accesses due to load or store instructions. Memory accesses are not counted if they are caused by any of the following actions: • Instruction fetches • Cache maintenance instructions • Translation table walks or prefetches This event counts the sum of MEM_ACCESS_RD and MEM_ACCESS_WR.
0x0014	L1I_CACHE	L1 instruction cache access. This event counts any instruction fetch that accesses the L1 instruction cache. The following instructions are not counted: • Cache maintenance instructions • Non-cacheable accesses
0x0015	L1D_CACHE_WB	L1 data cache Write-Back. This event counts any write-back of data from the L1 data cache to the L2 cache or the L3 cache. The event counts both victim line evictions and snoops, including cache maintenance operations. The following actions are not counted: • Invalidations that do not result in data being transferred out of the L1 cache • Full-line writes that write to L2 cache without writing L1 cache, such as write-streaming mode.
0x0016	L2D_CACHE	Level 2 data cache access. If the complex is configured with a per-complex L2 cache, this event counts: • Any transaction from the L1 cache that looks up in the L2 cache • Any write-back from the L1 cache to the L2 cache Snoops from outside the complex and cache maintenance operations are not counted. If the complex is not configured with a per-complex L2 cache, this event counts the cluster cache event, as defined by L3D_CACHE. If neither a per-complex cache or a cluster cache are configured, this event is not implemented.
0x0017	L2D_CACHE_REFILL	Level 2 data cache refill. If the complex is configured with a per-complex L2 cache, this event counts any Cacheable transaction from L1 that causes data to be read from outside the complex. L2 cache refills that are caused by stashes into the L2 cache are not counted. If the complex is not configured with a per-complex L2 cache, this event is not implemented.

Event number	Event mnemonic	Description
0x0018	L2D_CACHE_WB	Level 2 data cache Write-Back. If the complex is configured with a per-complex L2 cache, this event counts any write-back of data from the L2 cache to a location outside the complex. The event includes snoops to the L2 cache that return data, regardless of whether they cause an invalidation. Invalidations from the L2 cache that do not write data outside of the complex and snoops that return data from the L1 cache are not counted. If the complex is not configured with a per-complex L2 cache, this event is not implemented.
0x0019	BUS_ACCESS	Bus access. This event counts for every beat of data that is transferred over the data channels between the complex and the *DynamIQ™ Shared Unit-110* (DSU-110). If both read and write data beats are transferred on a given cycle, this event is counted twice on that cycle. This event counts the sum of BUS_ACCESS_RD and BUS_ACCESS_WR.
0x001A	MEMORY_ERROR	Local memory error. This event counts any correctable or uncorrectable memory error (ECC or parity) in the protected core RAMs.
0x001B	INST_SPEC	Operation Speculatively executed. This event counts issued instructions, including instructions that are later flushed due to mis-speculation.
0x001C	TTBR_WRITE_RETIRED	Instruction architecturally executed, Condition code check pass, write to TTBR. This event only counts writes to TTBR0_EL1 and TTBR1_EL1.
0x001D	BUS_CYCLES	Bus cycles. This event duplicates CPU_CYCLES. This event is not exported to the ETM.
0x001E	CHAIN	Odd performance counter chain mode. This event is not exported to the ETM.
0x0020	L2D_CACHE_ALLOCATE	Level 2 data cache allocation without refill. If the complex is configured with a per-complex L2 cache, this event counts any full cache line write into the L2 cache that does not cause a linefill. The event includes write-backs from L1 to L2 and full-line writes that do not allocate into the L1 cache. If the complex is not configured with a per-complex L2 cache, this event is not implemented.
0x0021	BR_RETIRED	Instruction architecturally executed, branch.
0x0022	BR_MIS_PRED_RETIRED	Instruction architecturally executed, mispredicted branch. The counter counts all instructions counted by BR_RETIRED that were not correctly predicted.
0x0023	STALL_FRONTEND	No operation issued due to the frontend. The counter counts on any cycle when no operations are issued due to the instruction queue being empty.

Event number	Event mnemonic	Description
`0x0024`	STALL_BACKEND	No operation issued due to the backend. The counter counts on any cycle when no operations are issued due to a pipeline stall.
`0x0025`	L1D_TLB	Level 1 data TLB access. This event counts any load or store operation that accesses the L1 data 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.
`0x0026`	L1I_TLB	Level 1 instruction TLB access. This event counts any instruction fetch that accesses the L1 instruction TLB. This event counts regardless of whether the MMU is enabled.
`0x002B`	L3D_CACHE	Attributable level 3 unified cache access. If the complex is configured with a per-complex L2 cache and the cluster is configured with an L3 cache, this event counts for any cacheable read transaction returning data from the DSU-110, or for any cacheable write to the DSU-110. If either the complex is configured without a per-complex L2 or the cluster is configured without an L3 cache, this event is not implemented.
`0x002D`	L2D_TLB_REFILL	Attributable Level 2 data TLB refill. This event counts on any refill of the L2 TLB, caused by either an instruction or data access. This event does not count if the MMU is disabled.
`0x002F`	L2D_TLB	Attributable Level 2 data or unified TLB access. This event counts on any access to the L2 TLB that is caused by a refill of any of the L1 TLBs. This event does not count if the MMU is disabled.
`0x0034`	DTLB_WALK	Access to data TLB that caused a translation table walk. This event counts on any data access that causes L2D_TLB_REFILL to count.
`0x0035`	ITLB_WALK	Access to instruction TLB that caused a translation table walk. This event counts on any instruction access that causes L2D_TLB_REFILL to count.
`0x0036`	LL_CACHE_RD	Last level cache access, read. If IMP_CPUECTLR_EL1.EXTLLC is set, this event counts any cacheable read transaction that returns a data source of "interconnect cache". If IMP_CPUECTLR_EL1.EXTLLC is not set, this event is a duplicate of the L*D_CACHE_RD event corresponding to the last level of cache implemented in the cluster. That is: • L3D_CACHE_RD, if both per-complex L2 cache and cluster L3 cache are implemented • L2D_CACHE_RD, if only one of these caches are implemented • L1D_CACHE_RD, if neither of these caches are implemented

Event number	Event mnemonic	Description
`0x0037`	LL_CACHE_MISS_RD	Last level cache miss, read. If IMP_CPUECTLR_EL1.EXTLLC is set, this event counts any cacheable read transaction that returns a data source of "DRAM", "remote", or "inter-cluster peer". If IMP_CPUECTLR_EL1.EXTLLC is not set, this event is a duplicate of the event that corresponds to the last level of cache implemented in the cluster. Therefore, this event is a duplicate of: • L3D_CACHE_REFILL_RD, if both per-complex L2 cache and cluster L3 cache are implemented • L2D_CACHE_REFILL_RD, if only one is implemented • L1D_CACHE_REFILL_RD, if neither is implemented
`0x0038`	REMOTE_ACCESS_RD	Access to another socket in a multi-socket system, read. This event counts any read transaction that returns a data source of "remote".
`0x0039`	L1D_CACHE_LMISS_RD	Level 1 data cache long-latency read miss. This event counts each memory read access counted by L1D_CACHE that incurs additional latency because it returns data from outside the L1 data or unified cache of this *Processing Element* (PE).
`0x003A`	OP_RETIRED	Micro-operation architecturally executed This event counts each operation counted by OP_SPEC that would be executed in a Simple sequential execution of the program.
`0x003B`	OP_SPEC	Micro-operation Speculatively executed This event counts the number of operations executed by the core, including those that are executed speculatively and would not be executed in a Simple sequential execution of the program.
`0x003C`	STALL	No operation sent for execution The counter counts every Attributable cycle on which no Attributable instruction or operation was sent for execution on this core.
`0x003D`	STALL_SLOT_BACKEND	No operation sent for execution on a Slot due to the backend The counter counts each Slot counted by STALL_SLOT where no Attributable instruction or operation was sent for execution because the backend is unable to accept one of: • The instruction operation available for the PE on the Slot • Any operations on the Slot
`0x003E`	STALL_SLOT_FRONTEND	No operation sent for execution on a Slot due to the frontend The counter counts each Slot counted by STALL_SLOT where no Attributable instruction or operation was sent for execution because there was no Attributable instruction or operation available to issue from the PE from the frontend for the Slot.
`0x003F`	STALL_SLOT	No operation sent for execution on a Slot The counter counts on each Attributable cycle the number of instruction or operation Slots that were not occupied by an instruction or operation Attributable to the PE.

Event number	Event mnemonic	Description
`0x0040`	L1D_CACHE_RD	Level 1 data cache access, read. This event counts any load 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_RD event causes this event to count The following instructions are not counted: • Cache maintenance instructions and prefetches • Non-cacheable accesses
`0x4005`	STALL_BACKEND_MEM	Memory stall cycles The counter is identical to STALL_BACKEND_MEM in the AMUv1 architecture.
`0x4006`	L1I_CACHE_LMISS	Level 1 instruction cache long-latency read miss The counter counts each access counted by L1I_CACHE that incurs more latency because it returns instructions from outside the L1 instruction cache.
`0x4009`	L2D_CACHE_LMISS_RD	Level 2 data cache long-latency read miss The counter counts each memory read access counted by L2D_CACHE that incurs more latency because it returns data from outside the L2 data cache or the unified cache of the core.
`0x400B`	L3D_CACHE_LMISS_RD	Level 3 data cache long-latency read miss The counter counts each memory read access counted by L3D_CACHE that incurs more latency because it returns data from outside the L3 data or unified cache of the core.
`0x400C`	TRB_WRAP	Trace buffer current write pointer wrapped The event is generated each time the current write pointer is wrapped to the base pointer.
`0x400D`	PMU_OVFS	PMU overflow, counters accessible to EL1 and EL0 The event is generated each time an event causes a PMEVCTNR<n>_EL1 counter overflow when PMINTENSET_EL1[n] is set to 1, for each implemented PMU counter n in the range 0 <= n < UInt(MDCR_EL2.HPMN), and the Cycle Counter (n = 31). **Note:** This event is only exported to the ETM and is not visible to the PMU.
`0x400E`	TRB_TRIG	Trace buffer Trigger Event The event is generated when a Trace Buffer Extension Trigger Event occurs.
`0x400F`	PMU_HOVFS	PMU overflow, counters reserved for use by EL2 **Note:** This event is only exported to the ETM and is not visible to the PMU.
`0x4010`	TRCEXTOUT0	Trace unit external outputs 0-3 The trace unit outputs 0-3 are connected to trigger input `<n>`, for `<n>` = 4 to 7 **Note:** These events are not exported to the ETM.
`0x4011`	TRCEXTOUT1	
`0x4012`	TRCEXTOUT2	
`0x4013`	TRCEXTOUT3	

Event number	Event mnemonic	Description
0x4018	CTI_TRIGOUT4	Cross Trigger Interface output trigger <n>, for <n> = 4 to 7 The event is generated each time an event is signaled on CTI output trigger <n>.
0x4019	CTI_TRIGOUT5	
0x401A	CTI_TRIGOUT6	
0x401B	CTI_TRIGOUT7	
0x4020	LDST_ALIGN_LAT	Access with additional latency from alignment The counter counts each access counted by MEM_ACCESS that incurred more latency because of the alignment of the address and the size of data being accessed.
0x4021	LD_ALIGN_LAT	Load with additional latency from alignment The counter counts each memory-read access counted by LDST_ALIGN_LAT.
0x4022	ST_ALIGN_LAT	Store with additional latency from alignment The counter counts each memory-write access counted by LDST_ALIGN_LAT.
0x4024	MEM_ACCESS_CHECKED	Checked data memory access The counter counts each memory access counted by MEM_ACCESS that is Tag Checked by the *Memory Tagging Extension* (MTE).
0x4025	MEM_ACCESS_CHECKED_RD	Checked data memory access, read The counter counts each memory-read access counted by MEM_ACCESS_CHECKED.
0x4026	MEM_ACCESS_CHECKED_WR	Checked data memory access, write The counter counts each memory-write access counted by MEM_ACCESS_CHECKED.
0x8002	SVE_INST_RETIRED	Instruction architecturally executed *Scalable Vector Extension* (SVE) The counter counts architecturally executed SVE instructions.
0x8006	SVE_INST_SPEC	SVE Operations speculatively executed The counter counts speculatively executed operations due to SVE instructions.
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
0x80E3	ASE_SVE_INT8_SPEC	Advanced SIMD and SVE 8-bit integer operation speculatively executed
0x80E7	ASE_SVE_INT16_SPEC	Advanced SIMD and SVE 16-bit integer operation speculatively executed
0x80EB	ASE_SVE_INT32_SPEC	Advanced SIMD and SVE 32-bit integer operation speculatively executed
0x80EF	ASE_SVE_INT64_SPEC	Advanced SIMD and SVE 64-bit integer operation speculatively executed

17.1.2 Arm recommended IMPLEMENTATION DEFINED performance monitors events

The Cortex®-A510 core *Performance Monitoring Unit* (PMU) collects Arm recommended **IMPLEMENTATION DEFINED** performance monitors events.

Arm recommends using the **IMPLEMENTATION DEFINED** event numbers for specific events as described in the following table. However, Arm does not define these events as rigorously as the events in the architectural and microarchitectural event lists. For your specific implementation, you might choose to:

- Not use some of these event numbers.
- Modify the definition of an event to better correspond to your implementation.

The following table shows the Arm recommended **IMPLEMENTATION DEFINED** PMU events. These events are not exported to the *Embedded Trace Macrocell* (ETM).

Table 17-2: Arm recommended IMPLEMENTATION DEFINED PMU events

Event number	Event mnemonic	Event name
`0x0041`	L1D_CACHE_WR	L1 data cache access, write. Counts any store operation that looks up in the L1 data cache. In particular, any access that could count the L1D_CACHE_REFILL event causes this event to count. Cache maintenance instructions, Non-cacheable accesses, and prefetches are not counted.
`0x0042`	L1D_CACHE_REFILL_RD	L1 data cache refill, read. This event counts any load operation or translation table walk access that causes data to be read from outside the L1 data cache. The event includes accesses that do not allocate into the L1 cache. Cache maintenance instructions, Non-cacheable accesses, and prefetches are not counted.
`0x0043`	L1D_CACHE_REFILL_WR	L1 data cache refill, write. This event counts any store operation that causes data to be read from outside the L1 data cache, including accesses that do not allocate into the L1 cache. The following instructions are not counted: • Cache maintenance instructions and prefetches • Stores of an entire cache line, even if they make a coherency request outside the L1 cache • Partial cache line writes that do not allocate into the L1 cache • Non-cacheable accesses
`0x0044`	L1D_CACHE_REFILL_INNER	L1 data cache refill, inner. This event counts any L1 data cache linefill, as counted by L1D_CACHE_REFILL, that hits in the L2 cache, L3 cache, or another core in the cluster.
`0x0045`	L1D_CACHE_REFILL_OUTER	L1 data cache refill, outer. This event counts any L1 data cache linefill, as counted by L1D_CACHE_REFILL, that does not hit in the L2 cache, L3 cache, or another core in the cluster, and instead obtains data from outside the cluster.

Event number	Event mnemonic	Event name
0x0050	L2D_CACHE_RD	L2 cache access, read. If the complex is configured with a per-complex L2 cache, this event counts any read transaction from the L1 cache that looks up in the L2 cache. Snoops from outside the complex are not counted. If the complex is configured without a per-complex L2 cache, this event counts the cluster cache event, as defined by L3D_CACHE_RD. If neither a per-complex cache or a cluster cache is configured, this event is not implemented.
0x0051	L2D_CACHE_WR	L2 cache access, write. If the complex is configured with a per-complex L2 cache, this event counts any write transaction from the L1 cache that looks up in the L2 cache or any write-back from L1 cache that allocates into the L2 cache. Snoops from outside the complex are not counted. If the complex is configured without a per-complex L2 cache, this event counts the cluster cache event, as defined by L3D_CACHE_WR. If neither a per-complex cache or a cluster cache is configured, this event is not implemented.
0x0052	L2D_CACHE_REFILL_RD	L2 cache refill, read. If the complex is configured with a per-complex L2 cache, this event counts any cacheable read transaction from L1 cache that causes data to be read from outside the complex. L2 cache refills caused by stashes into L2 are not counted. Transactions such as ReadUnique are counted here as read transactions, even though they can be generated by store instructions. If the complex is configured without a per-complex L2 cache, this event counts the cluster cache event, as defined by L3D_CACHE_REFILL_RD. If neither a per-complex cache or a cluster cache is configured, this event is not implemented.
0x0053	L2D_CACHE_REFILL_WR	L2 cache refill, write. If the complex is configured with a per-complex L2 cache, this event counts any write transaction from L1 cache that causes data to be read from outside the complex. L2 cache refills caused by stashes into L2 are not counted. Transactions such as ReadUnique are not counted as write transactions. If the complex is configured without a per-core L2 cache, this event is not implemented.
0x0060	BUS_ACCESS_RD	Bus access, read. This event counts for every beat of data that is transferred over the read data channel between the complex and the *DynamIQ™ Shared Unit-110* (DSU-110).
0x0061	BUS_ACCESS_WR	Bus access, write. This event counts for every beat of data that is transferred over the write data channel between the complex and the DSU-110.

Event number	Event mnemonic	Event name
0x0066	MEM_ACCESS_RD	Data memory access, read. This event counts memory accesses due to load instructions. The following actions are not counted: • Instruction fetches • Cache maintenance instructions • Translation table walks • Prefetches
0x0067	MEM_ACCESS_WR	Data memory access, write. This event counts memory accesses due to store instructions. The following actions are not counted: • Instruction fetches • Cache maintenance instructions • Translation table walks • Prefetches
0x0070	LD_SPEC	Operation speculatively executed, load.
0x0071	ST_SPEC	Operation speculatively executed, store.
0x0072	LDST_SPEC	Operation speculatively executed, load or store. This event counts the sum of LD_SPEC and ST_SPEC.
0x0073	DP_SPEC	Operation speculatively executed, integer data processing.
0x0074	ASE_SPEC	Operation speculatively executed, Advanced SIMD instruction.
0x0075	VFP_SPEC	Operation speculatively executed, floating-point instruction.
0x0076	PC_WRITE_SPEC	Operation speculatively executed, software change of the Program Counter.
0x0077	CRYPTO_SPEC	Operation speculatively executed, Cryptographic instruction.
0x0078	BR_IMMED_SPEC	Branch speculatively executed, immediate branch. This event duplicates BR_IMMED_RETIRED.
0x0079	BR_RETURN_SPEC	Branch speculatively executed, procedure return.
0x007A	BR_INDIRECT_SPEC	Branch speculatively executed, indirect branch.
0x0086	EXC_IRQ	Exception taken, IRQ.
0x0087	EXC_FIQ	Exception taken, FIQ.
0x00A0	L3D_CACHE_RD	Attributable L3 unified cache access, read. This event counts for any cacheable read transaction returning data from the DSU-110. If either the complex is configured without a per-complex L2 cache or the cluster is configured without an L3 cache, this event is not implemented.
0x00A2	L3D_CACHE_REFILL_RD	Attributable L3 unified cache refill, read. If either the complex is configured without a per-complex L2 cache or the cluster is configured without an L3 cache, this event is not implemented.

17.1.3 IMPLEMENTATION DEFINED performance monitors events

The Cortex®-A510 core *Performance Monitoring Unit* (PMU) collects **IMPLEMENTATION DEFINED** events.

The following table shows the **IMPLEMENTATION DEFINED** performance monitors events. These events are not exported to the *Embedded Trace Macrocell* (ETM).

Table 17-3: Arm IMPLEMENTATION DEFINED PMU events

Event number	Event mnemonic	Event name
`0x00C1`	L2D_CACHE_REFILL_PREFETCH	L2 cache refill due to prefetch. If the complex is configured with a per-complex L2 cache, this event does not count. If the complex is configured without a per-complex L2 cache, this event counts the cluster cache event, as defined by L3D_CACHE_REFILL_PREFETCH. If neither a per-complex cache or a cluster cache is configured, this event is not implemented.
`0x00C2`	L1D_CACHE_REFILL_PREFETCH	L1 data cache refill due to prefetch. This event counts any linefills from the prefetcher that cause an allocation into the L1 data cache.
`0x00C3`	L2D_WS_MODE	L2 cache write streaming mode. This event counts for each cycle where the core is in write streaming mode and is not allocating writes into the L2 cache.
`0x00C4`	L1D_WS_MODE_ENTRY	L1 data cache entering write streaming mode. This event counts for each entry into write streaming mode.
`0x00C5`	L1D_WS_MODE	L1 data cache write streaming mode. This event counts for each cycle where the core is in write streaming mode and is not allocating writes into the L1 data cache.
`0x00C7`	L3D_WS_MODE	L3 cache write streaming mode. This event counts for each cycle where the core is in write streaming mode and is not allocating writes into the L3 cache.
`0x00C8`	LL_WS_MODE	Last level cache write streaming mode. This event counts for each cycle where the core is in write streaming mode and is not allocating writes into the system cache.
`0x00C9`	BR_COND_PRED	Predicted conditional branch executed. This event counts when any branch that the conditional predictor can predict is retired. This event still counts when branch prediction is disabled due to the *Memory Management Unit* (MMU) being off.
`0x00CA`	BR_INDIRECT_MIS_PRED	Indirect branch mispredicted. This event counts when any indirect branch that the *Branch Target Address Cache* (BTAC) can predict is retired and has mispredicted either the condition or the address. This event still counts when branch prediction is disabled due to the MMU being off.

Event number	Event mnemonic	Event name
0x00CB	BR_INDIRECT_ADDR_MIS_PRED	Indirect branch mispredicted due to address miscompare. This event counts when any indirect branch that the BTAC can predict is retired, was taken, correctly predicted the condition, and has mispredicted the address. This event still counts when branch prediction is disabled due to the MMU being off.
0x00CC	BR_COND_MIS_PRED	Conditional branch mispredicted. This event counts when any branch that the conditional predictor can predict is retired and has mispredicted the condition. This event still counts when branch prediction is disabled due to the MMU being off. Conditional indirect branches that correctly predict the condition but mispredict the address do not count.
0x00CD	BR_INDIRECT_ADDR_PRED	Indirect branch with predicted address executed. This event counts when any indirect branch that the BTAC can predict is retired, was taken, and correctly predicted the condition. This event still counts when branch prediction is disabled due to the MMU being off.
0x00CE	BR_RETURN_ADDR_PRED	Procedure return with predicted address executed. This event counts when any procedure return that the call-return stack can predict is retired, was taken, and correctly predicted the condition. This event still counts when branch prediction is disabled due to the MMU being off.
0x00CF	BR_RETURN_ADDR_MIS_PRED	Procedure return mispredicted due to address miscompare. This event counts when any procedure return that the call-return stack can predict is retired, was taken, correctly predicted the condition, and has mispredicted the address. This event still counts when branch prediction is disabled due to the MMU being off.
0x00D0	L2D_WALK_TLB	L2 TLB walk cache access. This event does not count if the MMU is disabled.
0x00D1	L2D_WALK_TLB_REFILL	L2 TLB walk cache refill. This event does not count if the MMU is disabled.
0x00D4	L2D_S2_TLB	L2 TLB IPA cache access. This event counts on each access to the IPA cache. If a single translation table walk needs to make multiple accesses to the IPA cache, each access is counted. If stage 2 translation is disabled, this event does not count.
0x00D5	L2D_S2_TLB_REFILL	L2 TLB IPA cache refill. This event counts on each refill of the IPA cache. If a single translation table walk needs to make multiple accesses to the IPA cache, each access that causes a refill is counted. If stage 2 translation is disabled, this event does not count.
0x00D6	L2D_CACHE_STASH_DROPPED	L2 cache stash dropped. This event counts on each stash request that is received from the interconnect or the *Accelerator Coherency Port* (ACP), that targets L2 cache and is dropped due to lack of buffer space to hold the request.
0x00E1	STALL_FRONTEND_CACHE	No operation issued due to the frontend, cache miss. This event counts every cycle that the *Data Processing Unit* (DPU) instruction queue is empty and there is an instruction cache miss being processed.
0x00E2	STALL_FRONTEND_TLB	No operation issued due to the frontend, TLB miss. This event counts every cycle that the DPU instruction queue is empty and there is an instruction L1 TLB miss being processed.
0x00E3	STALL_FRONTEND_PDERR	No operation issued due to the frontend, pre-decode error.

Event number	Event mnemonic	Event name
`0x00E4`	STALL_BACKEND_ILOCK	No operation issued due to the backend interlock. This event counts every cycle where the issue of an operation is stalled and there is an interlock. Stall cycles due to a stall in the Wr stage are excluded.
`0x00E5`	STALL_BACKEND_ILOCK_ADDR	No operation issued due to the backend, address interlock. This event counts every cycle where the issue of an operation is stalled and there is an interlock on an address operand. This type of interlock is caused by a load/store instruction waiting for data to calculate the address. Stall cycles due to a stall in the Wr stage are excluded.
`0x00E6`	STALL_BACKEND_ILOCK_VPU	No operation issued due to the backend, interlock, or the *Vector Processing Unit* (VPU). This event counts every cycle where there is a stall or an interlock that is caused by a VPU instruction. Stall cycles due to a stall in the Wr stage are excluded.
`0x00E7`	STALL_BACKEND_LD	No operation issued due to the backend, load. This event counts every cycle where there is a stall in the Wr stage due to a load.
`0x00E8`	STALL_BACKEND_ST	No operation issued due to the backend, store. This event counts every cycle where there is a stall in the Wr stage due to a store.
`0x00E9`	STALL_BACKEND_LD_CACHE	No operation issued due to the backend, load, cache miss. This event counts every cycle where there is a stall in the Wr stage due to a load that is waiting on data. The event counts for stalls that are caused by missing the cache or where the data is Non-cacheable.
`0x00EA`	STALL_BACKEND_LD_TLB	No operation issued due to the backend, load, TLB miss. This event counts every cycle where there is a stall in the Wr stage due to a load that misses in the L1 TLB.
`0x00EB`	STALL_BACKEND_ST_STB	No operation issued due to the backend, store, *Store Buffer* (STB) full. This event counts every cycle where there is a stall in the Wr stage because of a store operation that is waiting due to the STB being full.
`0x00EC`	STALL_BACKEND_ST_TLB	No operation issued due to the backend, store, TLB miss. This event counts every cycle where there is a stall in the Wr stage because of a store operation that has missed in the L1 TLB.
`0x00ED`	STALL_BACKEND_VPU_HAZARD	No operation issued due to the backend, VPU hazard. This event counts every cycle where the core stalls due to contention for the VPU with the other core.
`0x00EE`	STALL_SLOT_BACKEND_ILOCK	Issue slot not issued due to interlock. For each cycle, this event counts each dispatch slot that does not issue due to an interlock.

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®-A510 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 Armv8-A architecture profile*. The register descriptions provided in this manual describe whether each register is read/write or read-only.

17.4 Performance monitors register summary

The summary table provides an overview of performance monitors registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table 17-4: 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_EL0	3	C9	3	C12	0	See individual bit resets	64-bit	Performance Monitors Control Register
PMCEID0_EL0	3	C9	3	C12	6	See individual bit resets	64-bit	Performance Monitors Common Event Identification register 0
PMCEID1_EL0	3	C9	3	C12	7	See individual bit resets	64-bit	Performance Monitors Common Event Identification register 1

17.5 Memory-mapped PMU register summary

The summary table provides an overview of memory-mapped *Performance Monitoring Unit* (PMU) registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table 17-5: 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

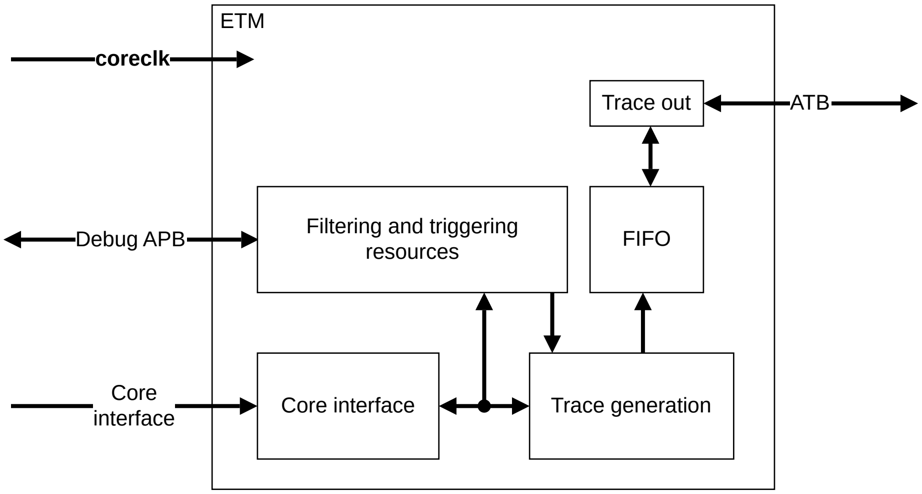
Offset	Name	Reset	Width	Description
0x610	PMSSSR	0x1	32-bit	PMU Snapshot Status Register
0x614	PMOVSSR	See individual bit resets	32-bit	PMU Overflow Status Snapshot Register
0x618	PMCCNTSR	See individual bit resets	64-bit	PMU Cycle Counter Snapshot Register
0x620	PMEVCNTSR0	See individual bit resets	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
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_EL0	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
0xFBC	PMDEVARCH	See individual bit resets	32-bit	Performance Monitors Device Architecture register
0xFC8	PMDEVID	See individual bit resets	32-bit	Performance Monitors Device ID register
0xFCC	PMDEVTYPE	See individual bit resets	32-bit	Performance Monitors Device Type register
0xFD0	PMPIDR4	See individual bit resets	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	PMPIDR0	See individual bit resets	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	PMPIDR1	See individual bit resets	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	PMPIDR2	See individual bit resets	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	PMPIDR3	See individual bit resets	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	PMCIDR0	See individual bit resets	32-bit	Performance Monitors Component Identification Register 0
0xFF4	PMCIDR1	See individual bit resets	32-bit	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	See individual bit resets	32-bit	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	See individual bit resets	32-bit	Performance Monitors Component Identification Register 3

18 Embedded Trace Extension support

The Cortex®-A510 core implements the *Embedded Trace Extension* (ETE) with an *Embedded Trace Macrocell* (ETM). The ETM performs real-time instruction flow tracing based on the ETE. The ETM 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 ETM:

Figure 18-1: ETM 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 ETM generates by filtering. For example, you can limit trace generation to a certain address range. The ETM supports other, more complicated, logic analyzer style filtering options. The ETM can also generate a trigger that is a signal to the Trace Capture Device to stop capturing trace.

FIFO

The ETM 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® Embedded Trace Extension* for more information.

18.1 Trace unit resources

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

The following table shows the *Embedded Trace Macrocell* (ETM) trace unit resources, and indicates which of these resources the A510 core implements.

Table 18-1: ETM trace unit resources implemented

Description	Configuration
Number of resource selection pairs implemented	8
Number of external input selectors implemented	4
Number of *Embedded Trace Extension* (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® Embedded Trace Extension* for more information.

18.2 Trace unit generation options

The Cortex®-A510 core *Embedded Trace Macrocell* (ETM) trace unit implements a set of generation options.

The following table shows the trace generation options that are implemented in the Cortex®-A510 core ETM trace unit.

Table 18-2: ETM trace unit generation options implemented

Description	Configuration
Instruction address size in bytes	8
Data address size in bytes	0, as the *Embedded Trace Extension* (ETE) does not implement data tracing
Data value size in bytes	0, as the ETE does not implement data tracing

Description	Configuration
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 P0 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	Implemented
Stall control support	Not implemented
Support for overflow avoidance	Not implemented
Support for using CONTEXTIDR_EL2 in *Virtual Machine IDentifier* (VMID) comparator	Implemented

See the *Arm® Embedded Trace Extension* for more information.

18.3 Reset the Embedded Trace Macrocell

The reset for the *Embedded Trace Macrocell* (ETM) is the same as a Cold reset for the core. In *TRace Buffer Extension* (TRBE) mode, a Warm reset keeps the TRBE disabled and therefore trace is not possible during Warm reset.

If the ETM trace unit is reset, then tracing stops until the ETM 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 Embedded Trace Macrocell registers

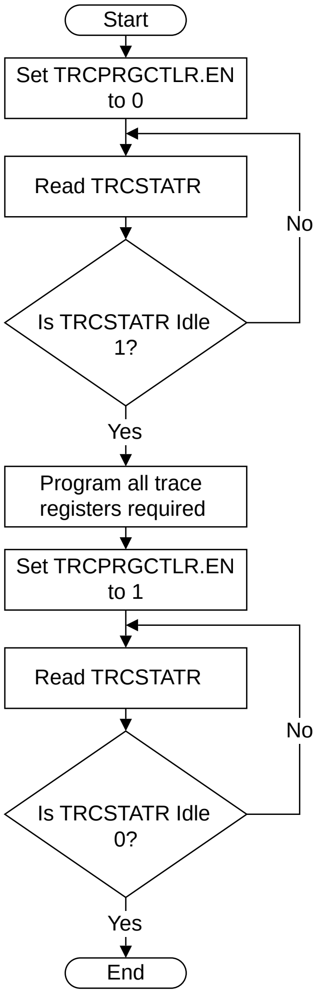
You program and read the *Embedded Trace Macrocell* (ETM) registers using either the Debug APB interface or the System register interface.

The core does not have to be in debug state when you program the ETM registers. When you program the ETM 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 ETM trace unit, use the TRCPRGCTLR.EN bit. See the *Arm® Embedded Trace Macrocell Architecture Specification* for more information about the following registers:

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

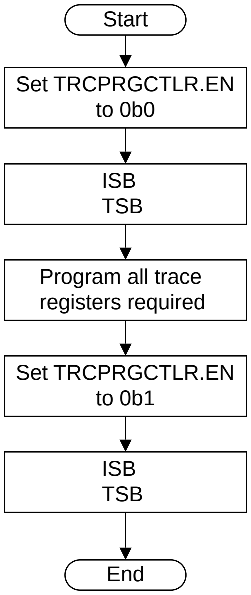
The following figure shows the flow for programming ETM registers using the Debug APB interface:

Figure 18-2: Programming ETM registers using the Debug APB interface



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

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



18.5 Embedded Trace Macrocell register interfaces

The Cortex®-A510 core supports an APB memory-mapped interface and a system register interface to *Embedded Trace Macrocell* (ETM) registers.

Register accesses differ depending on the ETM state. See the *Arm® Embedded Trace Extension* for information on the behaviors and access mechanisms.

18.6 Interaction with the Performance Monitoring Unit and Debug

The *Embedded Trace Macrocell* (ETM) interacts with the *Performance Monitoring Unit* (PMU) and it can access the PMU events.

Interaction with the PMU

The Cortex®-A510 core includes a PMU that enables events, such as cache misses and executed instructions, to be counted over time.

The PMU and ETM trace unit function together.

Use of PMU events by the ETM trace unit

The PMU architectural events are available to the ETM trace unit through the extended input facility. See the *Arm® Embedded Trace Extension* for more information about PMU events.

The ETM trace unit uses four extended external input selectors to access the PMU events. Each selector can independently select one of the PMU events, which is 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 ETM trace unit. The performance monitors events table describes the PMU events.

Related information

17 Performance Monitors Extension support on page 99

17.1 Performance monitors events on page 99

18.7 Embedded Trace Extension events

The Cortex®-A510 trace unit collects events from other units in the design and uses numbers to reference these events.

As part of the events mentioned in 17.1 Performance monitors events on page 99, the *Embedded Trace Extension* (ETE) events in the following table 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_HOVFS	PMU overflow, counters reserved for use by EL2

18.8 ETE register summary

The summary table provides an overview of memory-mapped *Embedded Trace Extension* (ETE) registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table 18-4: ETE register summary

Offset	Name	Reset	Width	Description
`0x18`	TRCAUXCTLR	`0x0`	32-bit	Auxiliary Control Register
`0x180`	TRCIDR8	See individual bit resets	32-bit	ID Register 8
`0x184`	TRCIDR9	`0x0`	32-bit	ID Register 9
`0x188`	TRCIDR10	`0x0`	32-bit	ID Register 10
`0x18C`	TRCIDR11	`0x0`	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

Offset	Name	Reset	Width	Description
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	0x0	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

19 Trace Buffer Extension support

The Cortex®-A510 core implements the *TRace Buffer Extension* (TRBE). The TRBE writes the program flow trace generated by the *Embedded Trace Macrocell* (ETM) trace unit directly to memory. The TRBE is programmed through System registers.

When enabled, the TRBE can:

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

When disabled, the TRBE ignores trace data and the ETM 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.

19.2 Trace buffer register interface

The Cortex®-A510 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 Armv8, for Armv8-A architecture profile* for information on the behaviors and access mechanisms.

19.3 TRBE register summary

The summary table provides an overview of *TRace Buffer Extension* (TRBE) registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table 19-1: TRBE register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
TRBIDR_EL1	3	C9	0	C11	7	See individual bit resets	64-bit	Trace Buffer ID Register

20 Activity Monitors Extension support

The Cortex®-A510 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®-A510 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®-A510 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 Armv8-A architecture profile* for information on the memory mapping of these registers.

Access enable bit

The access enable bit AMUSERENR_EL0.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_EL0.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 Armv8-A architecture profile*.

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®-A510 core instance number in the DSU-110 DynamIQ™ cluster.

20.2 Activity monitors counters

The Cortex®-A510 core implements seven activity monitors counters that map to specific *Activity Monitoring Unit* (AMU) events.

Each of the counters 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.
- 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®-A510 core are fixed 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	INSTR_RETIRED	`0x0008`	Instruction architecturally executed Increments for every instruction that is executed architecturally, including instructions that fail their condition code check
AMEVCNTR03	STALL_BACKEND_MEM	`0x4005`	Memory stall cycles Increments for each cycle 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	GEAR0_MPMM_ATHR_EXCEEDED	`0x0300`	*Maximum Power Mitigation System* (MPMM) Gear 0 Increments for each period where core activity is above the throttling threshold for gear 0 Reserved

Activity monitor counter <n>	Event	Event number	Description
AMEVCNTR11	GEAR1_MPMM_ATHR_EXCEEDED	`0x0301`	MPMM Gear 1 Increments for each period where core activity is above the throttling threshold for gear 1 Reserved
AMEVCNTR12	GEAR2_MPMM_ATHR_EXCEEDED	`0x0302`	MPMM Gear 2 Increments for each period where core activity is above the throttling threshold for gear 2 Reserved

Related information

5.3.1 Maximum Power Mitigation Mechanism on page 45

20.4 Activity monitors register summary

The summary table provides an overview of activity monitors registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table 20-2: Activity monitors register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
AMEVTYPER10_EL0	3	C13	3	C14	0	See individual bit resets	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_EL0	3	C13	3	C14	1	See individual bit resets	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_EL0	3	C13	3	C14	2	See individual bit resets	64-bit	Activity Monitors Event Type Registers 1
AMCFGR_EL0	3	C13	3	C2	1	See individual bit resets	64-bit	Activity Monitors Configuration Register
AMCGCR_EL0	3	C13	3	C2	2	See individual bit resets	64-bit	Activity Monitors Counter Group Configuration Register
AMEVTYPER00_EL0	3	C13	3	C6	0	See individual bit resets	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_EL0	3	C13	3	C6	1	See individual bit resets	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_EL0	3	C13	3	C6	2	See individual bit resets	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_EL0	3	C13	3	C6	3	See individual bit resets	64-bit	Activity Monitors Event Type Registers 0

20.5 Memory-mapped AMU register summary

The summary table provides an overview of memory-mapped *Activity Monitoring Unit* (AMU) registers in the Cortex®-A510 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
0xCE0	AMCGCR	See individual bit resets	32-bit	Activity Monitors Counter Group Configuration Register
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

Appendix A IMPLEMENTATION DEFINED behaviors

The Cortex®-A510 core has certain **IMPLEMENTATION DEFINED** behaviors.

Exclusive monitor

The exclusive state machine includes the following **IMPLEMENTATION DEFINED** transitions:

- If the monitor is in the exclusive state, and a Store-Exclusive instruction accesses a different address, the instruction fails and does not update memory.
- If a normal store instruction accesses a different address, it does not affect the exclusive monitor.
- If a normal store instruction is executed from a different core to the same address, it clears the exclusive monitor.
- If a normal store instruction is executed from the same core, it does not clear the exclusive monitor.

Appendix B AArch64 registers

This appendix contains the descriptions for the Cortex®-A510 AArch64 registers.

B.1 Generic system control register summary

The summary table provides an overview of generic system control registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table B-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)
IMP_CPUACTLR_EL1	3	C15	0	C1	0	See individual bit resets.	64-bit	CPU Auxiliary Control Register
IMP_CPUACTLR2_EL1	3	C15	0	C1	1	See individual bit resets.	64-bit	CPU Auxiliary Control Register 2
IMP_CMPXACTLR_EL1	3	C15	0	C1	3	See individual bit resets.	64-bit	Complex Auxiliary Control Register
IMP_CPUECTLR_EL1	3	C15	0	C1	4	See individual bit resets.	64-bit	CPU Extended Control Register
IMP_CMPXECTLR_EL1	3	C15	0	C1	7	See individual bit resets.	64-bit	Complex Extended Control Register
IMP_CPUPWRCTLR_EL1	3	C15	0	C2	7	See individual bit resets.	64-bit	CPU Power Control Register
IMP_ATCR_EL2	3	C15	4	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register
IMP_AVTCR_EL2	3	C15	4	C7	1	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register
IMP_ATCR_EL1	3	C15	5	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register
IMP_ATCR_EL3	3	C15	6	C7	0	See individual bit resets.	64-bit	CPU Auxiliary Translation Control Register

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
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)

B.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 B-1: AArch64_aidr_el1 bit assignments

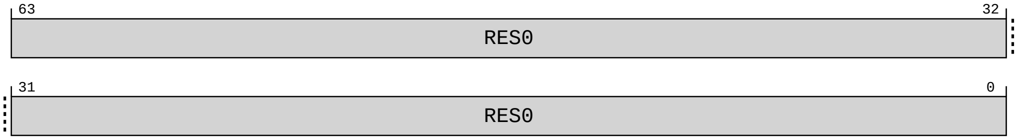


Table B-2: AIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == 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;
```

B.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 B-2: AArch64_actlr_el1 bit assignments

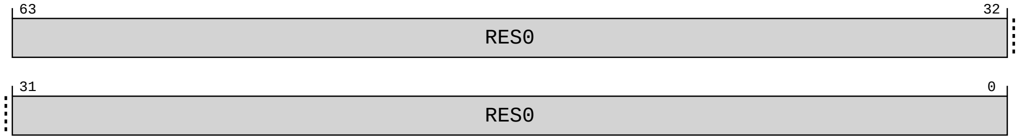


Table B-4: ACTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

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);
    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);
    else
        ACTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    ACTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
```

```
    ACTLR_EL1 = X[t];
```

B.1.3 ACTLR_EL2, Auxiliary Control Register (EL2)

Provides **IMPLEMENTATION DEFINED** configuration and control options for EL2.

Note

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

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure B-3: AArch64_actlr_el2 bit assignments

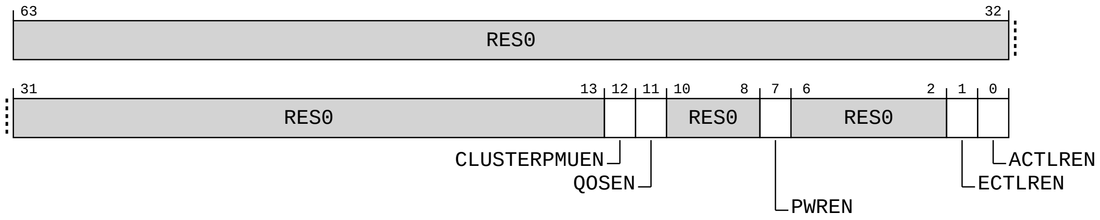


Table B-7: ACTLR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:13]	RES0	Reserved	0b0
[12]	CLUSTERPMUEN	Cluster PMU Registers enable. Traps EL1 writes to **IMPLEMENTATION DEFINED** cluster PMU registers to EL2. Possible values of this bit are: **0** This control causes writes to IMP_CLUSTERPM* at EL1 to be trapped. **1** This control does not cause any instructions to be trapped.	0b0

Bits	Name	Description	Reset
[11]	QOSEN	Cluster Bus QoS Registers enable. Traps EL1 writes to AArch64-IMP_CLUSTERBUSQOS_EL1 to EL2. Possible values of this bit are: **0** This control causes writes to AArch64-IMP_CLUSTERBUSQOS_EL1 at EL1 to be trapped. **1** This control does not cause any instructions to be trapped.	0b0
[10:8]	RES0	Reserved	0b0
[7]	PWREN	Power Control Registers enable. Traps EL1 writes to **IMPLEMENTATION DEFINED** power control registers to EL2. Possible values of this bit are: **0** This control causes writes to AArch64-IMP_CPUPWRCTLR_EL1, AArch64-IMP_CLUSTERPWRCTLR_EL1, AArch64-IMP_CLUSTERPWRDN_EL1 and IMP_CLUSTERL3*_EL1 at EL1 to be trapped. **1** This control does not cause any instructions to be trapped.	0b0
[6:2]	RES0	Reserved	0b0
[1]	ECTLREN	Extended Control Registers enable. Traps EL1 writes to AArch64-IMP_CPUECTLR_EL1, AArch64-IMP_CMPXECTLR_EL1 and AArch64-IMP_CLUSTERECTLR_EL1 to EL2. Possible values of this bit are: **0** This control causes writes to AArch64-IMP_CPUECTLR_EL1, AArch64-IMP_CMPXECTLR_EL1 and AArch64-IMP_CLUSTERECTLR_EL1 at EL1 to be trapped. **1** This control does not cause any instructions to be trapped.	0b0
[0]	ACTLREN	Auxiliary Control Registers enable. Traps EL1 writes to AArch64-IMP_CPUACTLR_EL1, AArch64-IMP_CPUACTLR2_EL1, AArch64-IMP_CMPXACTLR_EL1 and AArch64-IMP_CLUSTERACTLR_EL1 to EL2. Possible values of this bit are: **0** This control causes writes to AArch64-IMP_CPUACTLR_EL1, AArch64-IMP_CPUACTLR2_EL1, AArch64-IMP_CMPXACTLR_EL1 and AArch64-IMP_CLUSTERACTLR_EL1 at EL1 to be trapped. **1** This control does not cause any instructions to be trapped.	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;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    return ACTLR_EL2;
elsif PSTATE.EL == EL3 then
    return ACTLR_EL2;
```

MSR ACTLR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    ACTLR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    ACTLR_EL2 = X[t];
```

B.1.4 HACR_EL2, Hypervisor Auxiliary Control Register

Controls trapping to EL2 of **IMPLEMENTATION DEFINED** aspects of EL1 or EL0 operation.

Note

Arm recommends that the values in this register do not cause unnecessary traps to EL2 when AArch64-HCR_EL2.{E2H, TGE} == {1, 1}.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

`0x0`

Bit descriptions

Figure B-4: AArch64_hacr_el2 bit assignments

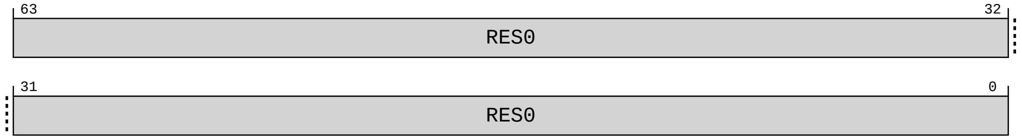


Table B-10: HACR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

Access

MRS <Xt>, HACR_EL2

<systemreg>	op0	op1	CRn	CRm	op2
HACR_EL2	0b11	0b100	0b0001	0b0001	0b111

MSR HACR_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
HACR_EL2	0b11	0b100	0b0001	0b0001	0b111

Accessibility

MRS <Xt>, HACR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    return HACR_EL2;
elsif PSTATE.EL == EL3 then
    return HACR_EL2;
```

MSR HACR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    HACR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    HACR_EL2 = X[t];
```

B.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 B-5: AArch64_actlr_el3 bit assignments

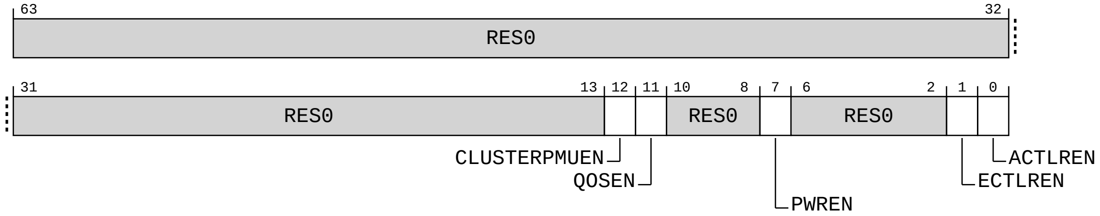


Table B-13: ACTLR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:13]	RES0	Reserved	0b0
[12]	CLUSTERPMUEN	Cluster PMU Registers enable. Traps EL1 and EL2 writes to **IMPLEMENTATION DEFINED** cluster PMU registers to EL3, subject to the exception prioritization rules. Possible values of this bit are: **0** This control causes writes to IMP_CLUSTERPM* at EL1 and EL2 to be trapped to EL3, subject to the exception prioritization rules.. **1** This control does not cause any instructions to be trapped.	0b0
[11]	QOSEN	Cluster Bus QoS Registers enable. Traps EL1 and EL2 writes to AArch64-IMP_CLUSTERBUSQOS_EL1 to EL3, subject to the exception prioritization rules. Possible values of this bit are: **0** This control causes writes to AArch64-IMP_CLUSTERBUSQOS_EL1 at EL1 and EL2 to be trapped to EL3, subject to the exception prioritization rules. **1** This control does not cause any instructions to be trapped.	0b0
[10:8]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[7]	PWREN	Power Control Registers enable. Traps EL1 and EL2 writes to **IMPLEMENTATION DEFINED** power control registers to EL3, subject to the exception prioritization rules. Possible values of this bit are: **0** This control causes writes to AArch64-IMP_CPUPWRCTLR_EL1, AArch64-IMP_CLUSTERPWRCTLR_EL1, AArch64-IMP_CLUSTERPWRDN_EL1 and IMP_CLUSTERL3*_EL1 at EL1 and EL2 to be trapped to EL3, subject to the exception prioritization rules. **1** This control does not cause any instructions to be trapped.	`0b0`
[6:2]	RES0	Reserved	`0b0`
[1]	ECTLREN	Extended Control Registers enable. Traps EL1 and EL2 writes to AArch64-IMP_CPUECTLR_EL1, AArch64-IMP_CMPXECTLR_EL1 and AArch64-IMP_CLUSTERECTLR_EL1 to EL3, subject to the exception prioritization rules. Possible values of this bit are: **0** This control causes writes to AArch64-IMP_CPUECTLR_EL1, AArch64-IMP_CMPXECTLR_EL1 and AArch64-IMP_CLUSTERECTLR_EL1 at EL1 and EL2 to be trapped to EL3, subject to the exception prioritization rules. **1** This control does not cause any instructions to be trapped.	`0b0`
[0]	ACTLREN	Auxiliary Control Registers enable. Traps EL1 and EL2 writes to AArch64-IMP_CPUACTLR_EL1, AArch64-IMP_CPUACTLR2_EL1, AArch64-IMP_CMPXACTLR_EL1 and AArch64-IMP_CLUSTERACTLR_EL1 to EL3, subject to the exception prioritization rules. Possible values of this bit are: **0** This control causes writes to AArch64-IMP_CPUACTLR_EL1, AArch64-IMP_CPUACTLR2_EL1, AArch64-IMP_CMPXACTLR_EL1 and AArch64-IMP_CLUSTERACTLR_EL1 at EL1 and EL2 to be trapped to EL3, subject to the exception prioritization rules. **1** This control does not cause any instructions to be trapped.	`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];
```

B.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

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

AMAIR_EL2 is permitted to be cached in a TLB.

Figure B-6: AArch64_amair_el2 bit assignments

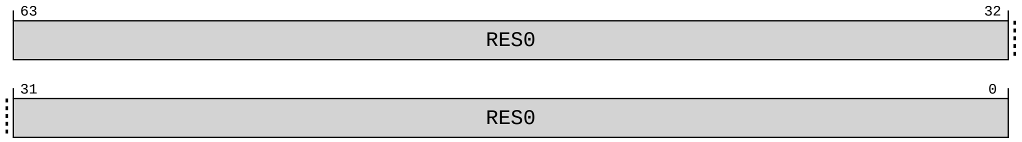


Table B-16: AMAIR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

Access

MRS <Xt>, AMAIR_EL2

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL2	0b11	0b100	0b1010	0b0011	0b000

MSR AMAIR_EL2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL2	0b11	0b100	0b1010	0b0011	0b000

MRS <Xt>, AMAIR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL1	0b11	0b000	0b1010	0b0011	0b000

MSR AMAIR_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL1	0b11	0b000	0b1010	0b0011	0b000

Accessibility

MRS <Xt>, AMAIR_EL2

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    return AMAIR_EL2;
elsif PSTATE.EL == EL3 then
    return AMAIR_EL2;
```

MSR AMAIR_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    AMAIR_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    AMAIR_EL2 = X[t];
```

MRS <Xt>, AMAIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return AMAIR_EL1;
```

```
elsif 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, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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];
```

B.1.7 LORID_EL1, LORegionID (EL1)

Indicates the number of LORegions and LORegion descriptors supported by the PE.

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 B-7: AArch64_lorid_el1 bit assignments

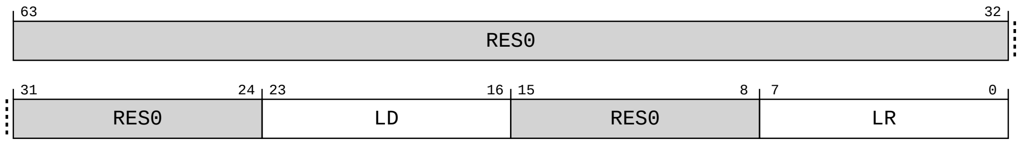


Table B-21: LORID_EL1 bit descriptions

Bits	Name	Description	Reset
[63:24]	RES0	Reserved	0b0
[23:16]	LD	Number of LORegion descriptors supported by the PE. This is an 8-bit binary number. **00000100** Four LOR descriptors are supported	
[15:8]	RES0	Reserved	0b0
[7:0]	LR	Number of LORegions supported by the PE. This is an 8-bit binary number. **Note:** If LORID_EL1 indicates that no LORegions are implemented, then LoadLOAcquire and StoreLORelease will behave as LoadAcquire and StoreRelease. **00000100** 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;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TLOR == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif SCR_EL3.TLOR == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return LORID_EL1;
elsif PSTATE.EL == EL2 then
    if SCR_EL3.TLOR == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return LORID_EL1;
elsif PSTATE.EL == EL3 then
    return LORID_EL1;
```

B.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 B-8: AArch64_amair_el1 bit assignments

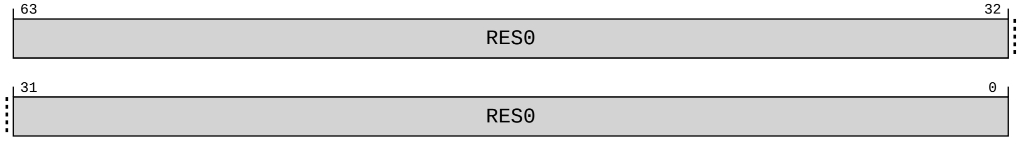


Table B-23: AMAIR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	`0b0`

Access

MRS <Xt>, AMAIR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL1	`0b11`	`0b000`	`0b1010`	`0b0011`	`0b000`

MSR AMAIR_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL1	`0b11`	`0b000`	`0b1010`	`0b0011`	`0b000`

MRS <Xt>, AMAIR_EL12

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL12	`0b11`	`0b101`	`0b1010`	`0b0011`	`0b000`

MSR AMAIR_EL12, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AMAIR_EL12	`0b11`	`0b101`	`0b1010`	`0b0011`	`0b000`

Accessibility

MRS <Xt>, AMAIR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return AMAIR_EL1;
elsif 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, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    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 <Xt>, AMAIR_EL12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    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, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == 1 then
        AMAIR_EL1 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == 1 then
        AMAIR_EL1 = X[t];
    else
        UNDEFINED;
```

B.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 B-9: AArch64_amair_el3 bit assignments

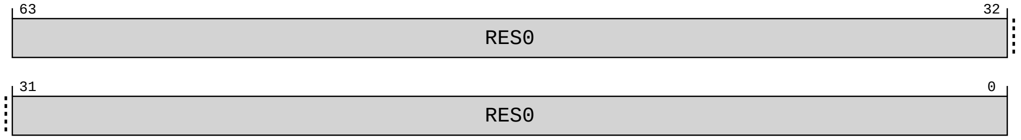


Table B-28: AMAIR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

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;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return AMAIR_EL3;
```

MSR AMAIR_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
    AMAIR_EL3 = X[t];
```

B.1.10 IMP_CPUACTLR_EL1, CPU Auxiliary 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 B-10: AArch64_imp_cpuactlr_el1 bit assignments

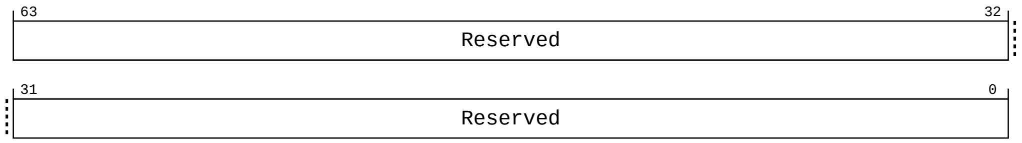


Table B-31: IMP_CPUACTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use	

Access

MRS <Xt>, S3_0_C15_C1_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_0	0b11	0b000	0b1111	0b0001	0b000

MSR S3_0_C15_C1_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_0	0b11	0b000	0b1111	0b0001	0b000

Accessibility

MRS <Xt>, S3_0_C15_C1_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_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];
```

B.1.11 IMP_CPUACTLR2_EL1, CPU Auxiliary 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 B-11: AArch64_imp_cpuactlr2_el1 bit assignments

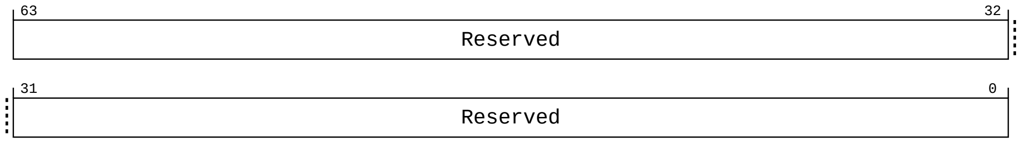


Table B-34: IMP_CPUACTLR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	Reserved	Reserved for Arm internal use	

Access

MRS <Xt>, S3_0_C15_C1_1

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_1	0b11	0b000	0b1111	0b0001	0b001

MSR S3_0_C15_C1_1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_1	0b11	0b000	0b1111	0b0001	0b001

Accessibility

MRS <Xt>, S3_0_C15_C1_1

```
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_CPUACTLR2_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CPUACTLR2_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUACTLR2_EL1;
```

MSR S3_0_C15_C1_1, <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_CPUACTLR2_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == 0 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CPUACTLR2_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CPUACTLR2_EL1 = X[t];
```

B.1.12 IMP_CMPXACTLR_EL1, Complex Auxiliary Control Register

This register contains control bits that affect the behavior of shared logic in a complex.

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 B-12: AArch64_imp_cmpxactlr_el1 bit assignments

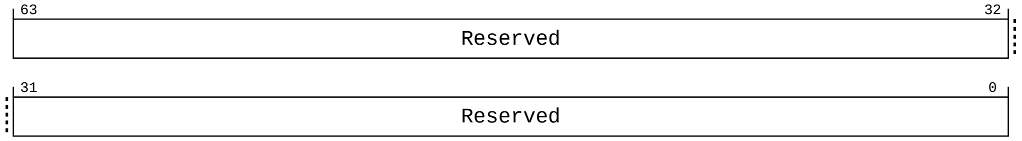


Table B-37: IMP_CMPXACTLR_EL1 bit descriptions

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_CMPXACTLR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CMPXACTLR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CMPXACTLR_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.ACTLREN == 0 then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ACTLREN == 0 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMPXACTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ACTLREN == 0 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMPXACTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CMPXACTLR_EL1 = X[t];
```

B.1.13 IMP_CPUECTLR_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 B-13: AArch64_imp_cpuectlr_el1 bit assignments

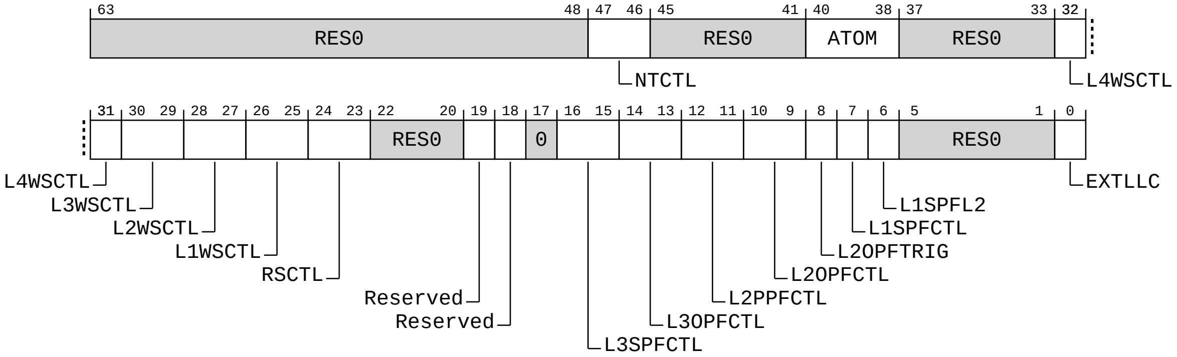


Table B-40: IMP_CPUECTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[47:46]	NTCTL	Transient/non-temporal L1 eviction control. **00** Transient/non-temporal lines evicted from the L1 cache skip L2 allocation, and allocate into the L3 cache as least-recently-used. **01** Transient/non-temporal lines evicted from the L1 cache allocate to the L2 as least-recently-used, and when evicted from the L2 allocate to the L3 as near-least-recently-used. **10** Transient/non-temporal clean lines evicted from the L1 cache are evicted without data. Dirty lines skip L2 allocation, and are allocated into the L3 cache if the line originally came from the L3 cache, otherwise are allocated into the SLC instead.	`0b00`
[45:41]	RES0	Reserved	`0b0`
[40:38]	ATOM	Atomic instruction handling policy **000** Atomic stores will be executed far unless they hit in a unique state in the L1 data cache, all other atomic instructions will be executed near. **001** All atomic instructions will be executed far unless they hit in a unique state in the L1 data cache. **010** All atomic instructions will be executed near. **011** All atomic instructions will be executed far. **100** Atomic stores will be executed far unless they hit in a unique state in the L1 data cache, all other atomic instructions will be executed near if they hit the L1 data cache, far otherwise.	`0b000`
[37:33]	RES0	Reserved	`0b0`
[32:31]	L4WSCTL	System cache write streaming threshold. Controls the threshold of the number of consecutive cache lines which are fully written without being read before stores are marked as Outer No Write-Allocate. **00** 512 cache lines. **01** 2048 cache lines. **10** 8191 cache lines. **11** Disable write streaming through system cache. All cache lines fetched due to stores will be marked as Outer Write-Allocate.	`0b00`

Bits	Name	Description	Reset
[30:29]	L3WSCTL	L3 write streaming threshold. Controls the threshold of the number of consecutive cache lines which are fully written without being read before stores stop causing L3 cache allocations. **00** 128 cache lines. **01** 1024 cache lines. **10** 4096 cache lines. **11** Disable write streaming through L3 cache. All cache lines fetched due to stores will allocate in L1, L2 or L3 caches.	0b00
[28:27]	L2WSCTL	L2 write streaming threshold. Controls the threshold of the number of consecutive cache lines which are fully written without being read before stores stop causing L2 cache allocations. **00** 16 cache lines. **01** 128 cache lines. **10** 512 cache lines. **11** Disable write streaming through L2 cache. All cache lines fetched due to stores will allocate in L1 or L2 caches.	0b00
[26:25]	L1WSCTL	L1 write streaming threshold. Controls the threshold of the number of consecutive cache lines which are fully written without being read before stores stop causing L1 cache allocations. **00** 4 cache lines. **01** 64 cache lines. **10** 128 cache lines. **11** Disable write streaming.	0b00
[24:23]	RSCTL	Read streaming aggressiveness control. **00** Normal operation. Clean read-streaming lines evicted from the L1 cache are evicted without data. Dirty lines skip L2 allocation, and are allocated into the L3 cache if the line originally came from the L3 cache, otherwise are allocated into the SLC instead. **01** Normal operation. Read-streaming lines are treated the same as transient/non-temporal lines. **11** Read streaming disabled.	0b00
[22:20]	RES0	Reserved	0b0
[19]	Reserved_19	Reserved for Arm internal use	
[18]	Reserved_18	Reserved for Arm internal use	

Bits	Name	Description	Reset
[17]	RES0	Reserved	0b0
[16:15]	L3SPFCTL	L3 cache stride prefetcher aggressiveness control. **00** Dynamic L3 stride prefetcher aggressiveness. **01** Conservative L3 stride prefetching. **10** Aggressive L3 stride prefetching.	0b00
[14:13]	L3OPFCTL	L3 cache offset prefetcher aggressiveness control. **00** Dynamic L3 offset prefetcher aggressiveness. **01** Conservative L3 offset prefetching. **10** Aggressive L3 offset prefetching. **11** L3 offset prefetching disabled.	0b00
[12:11]	L2PPFCTL	L2 cache pattern prefetcher aggressiveness control. **00** Very conservative L2 pattern prefetching. **01** Conservative L2 pattern prefetching. **11** Aggressive L2 pattern prefetching.	0b00
[10:9]	L2OPFCTL	L2 cache offset prefetcher aggressiveness control. **00** Dynamic L2 offset prefetcher aggressiveness. **01** Conservative L2 offset prefetching. **10** Very conservative L2 offset prefetching. **11** Most conservative L2 offset prefetching.	0b00
[8]	L2OPFTRIG	Offset prefetcher trigger control. **0** Trigger offset prefetcher based on pattern prefetcher. **1** Disable trigger of offset prefetcher based on pattern prefetcher.	0b0

Bits	Name	Description	Reset
[7]	L1SPFCTL	L1 cache stride prefetcher aggressiveness control. **0** Dynamic stride prefetcher aggressiveness. **1** Conservative stride prefetching.	0b0
[6]	L1SPFL2	Stride prefetcher cache level control. **0** Stride prefetcher prefetches into L1 and L3. **1** Stride prefetcher prefetches into L1 and L2.	0b0
[5:1]	RES0	Reserved	0b0
[0]	EXTLLC	Indicates that an external Last-level cache is present in the system, and that the DataSource field on the master CHI interface will indicate when data is returned from the LLC. Used to control how the LL_CACHE* PMU events count. **0** The last level cache in PMU events is within the cluster. **1** The last level cache in PMU events is outside the cluster.	0b0

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];
```

B.1.14 IMP_CMPXECTLR_EL1, Complex Extended Control Register

This register contains control bits that affect the behavior of shared logic in a complex.

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 B-14: AArch64_imp_cmpxectlr_el1 bit assignments

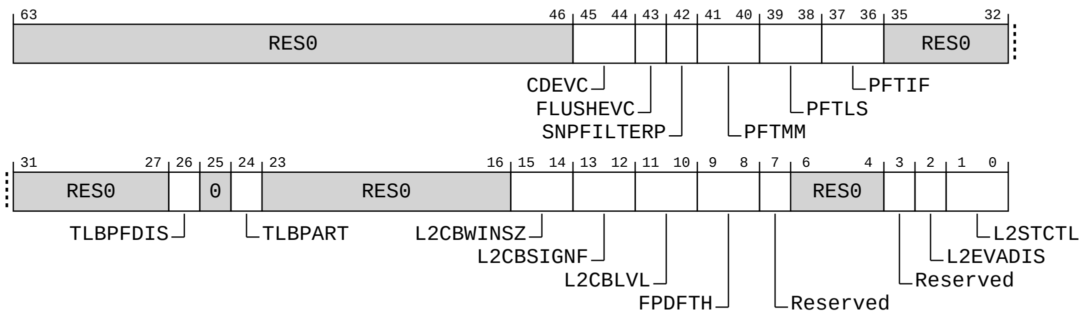


Table B-43: IMP_CMPXECTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:46]	RES0	Reserved	`0b0`

[45:44]	CDEVC	Downstream Cache Control
		`00`
		Disables sending data when clean cache-lines are evicted.
		`01`
		Enables sending WriteEvictFull transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data.
		`10`
		Enables sending WriteEvictOrEvict transactions when Unique Clean cache-lines are evicted. Shared Clean cache-line evictions do not send data.
		`11`
		Enables sending WriteEvictOrEvict transactions when Unique Clean or Shared Clean cache-lines are evicted.
			`0b11`

[43]	FLUSHEVC	Eviction Flush Control
		`0`
		Disables sending data when hardware cache flushes or DC CISW instructions evict a clean cache-line. .
		`1`
		Sending of data when hardware cache flushes or DC CISW instructions evict clean cache-lines is controlled by Downstream Cache Control. Sending of Evict transactions is controlled by SNPFILTERP.
			`0b0`

[42]	SNPFILTERP	Downstream Snoop Filter Present
		`0`
		Disables sending Evict transactions when clean cache-lines are evicted without data.
		`1`
		Enables sending Evict transactions when clean cache-lines are evicted without data.
			`0b1`

[41:40]	PFTMM	DRAM prefetch using PrefetchTgt transactions for table walk requests.
		`00`
		Disable PrefetchTgt generation for requests from the Memory Management unit (MMU).
		`01`
		Dynamically generate PrefetchTgt for requests from the MMU.
		`11`
		Always generate PrefetchTgt for requests from the MMU.
			`0b00`

[39:38]	PFTLS	DRAM prefetch using PrefetchTgt transactions for load and store requests.
		`00`
		Disable PrefetchTgt generation for requests from the Load-Store unit (LS).
		`01`
		Dynamically generate PrefetchTgt for requests from the LS.
		`11`
		Always generate PrefetchTgt for requests from the LS.
			`0b00`

Bits	Name	Description	Reset
[37:36]	PFTIF	DRAM prefetch using PrefetchTgt transactions for instruction fetch requests. **00** Disable PrefetchTgt generation for requests from the Instruction Fetch unit (IF). **01** Dynamically generate PrefetchTgt for requests from the IF. **11** Always generate PrefetchTgt for requests from the IF.	`0b00`
[35:27]	RES0	Reserved	`0b0`
[26]	TLBPFDIS	Disable L2 TLB prefetcher **0** The L2 TLB prefetcher is enabled. **1** The L2 TLB prefetcher is disabled.	`0b0`
[25]	RES0	Reserved	`0b0`
[24]	TLBPART	Partition L2 TLB allocations by core **When the complex contains two cores** **1** Core 0 can only allocate to ways 0-3 in the L2 TLB, and core 1 can only allocate to ways 4-7. Cores can hit entries allocated by either core.	
[23:16]	RES0	Reserved	`0b0`
[15:14]	L2CBWINSZ	Number of CBUSY responses in one sampling window. **00** 64 CBUSY responses per sampling window. **01** 128 CBUSY responses per sampling window. **10** 256 CBUSY responses per sampling window. **11** 512 CBUSY responses per sampling window.	`0b10`
[13:12]	L2CBSIGNF	Fraction of CBUSY responses in the sampling window necessary to be considered a valid sample of that CBUSY value. **00** 1/32 **01** 1/16 **10** 1/8 **11** 1/4	`0b01`

Bits	Name	Description	Reset
[11:10]	L2CBLVL	L2 internal CBUSY generation control. 00 Disable internal CBUSY generation. 01 Normal thresholds. 10 Conservative thresholds - throttles early. 11 Most conservative thresholds - throttles earlier.	0b01
[9:8]	FPDFTH	Prefetch data forwarding threshold. The value 0b11 disables prefetch data forwarding. 00 Default prefetch forwarding behaviour. 01 Faster prefetch forwarding timeout. 10 Immediate prefetch forwarding timeout (no waiting). 11 Prefetch forwarding is disabled.	0b00
[7]	Reserved_7	Reserved for Arm internal use	
[6:4]	RES0	Reserved	0b0
[3]	Reserved_3	Reserved for Arm internal use	
[2]	L2EVADIS	Disable L2 cache data RAM EVA accesses 0 Optimized evict/allocate accesses to L2 cache data RAMs using RAM EVA feature are enabled. 1 Optimized evict/allocate accesses to L2 cache data RAMs using RAM EVA feature are disabled.	0b0
[1:0]	L2STCTL	L2 cache stashing control 00 Stashes targeting L2 cache will allocate as if the line were brought in by a load. 01 Stashes targeting L2 cache will allocate and be marked as preferred targets for eviction. 10 Stashes targeting L2 cache will allocate as if the line were brought in by a load, but will only allocate to odd numbered cache ways. 11 Stashes targeting L2 cache will be ignored.	0b00

Access

MRS <Xt>, S3_0_C15_C1_7

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_7	0b11	0b000	0b1111	0b0001	0b111

MSR S3_0_C15_C1_7, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C1_7	0b11	0b000	0b1111	0b0001	0b111

Accessibility

MRS <Xt>, S3_0_C15_C1_7

```
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_CMPXECTLR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_CMPXECTLR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CMPXECTLR_EL1;
```

MSR S3_0_C15_C1_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.ECTLREN == 0 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif ACTLR_EL3.ECTLREN == 0 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMPXECTLR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if ACTLR_EL3.ECTLREN == 0 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        IMP_CMPXECTLR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_CMPXECTLR_EL1 = X[t];
```

B.1.15 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

See individual bit resets.

Bit descriptions

Figure B-15: AArch64_imp_cpupwrctlr_el1 bit assignments

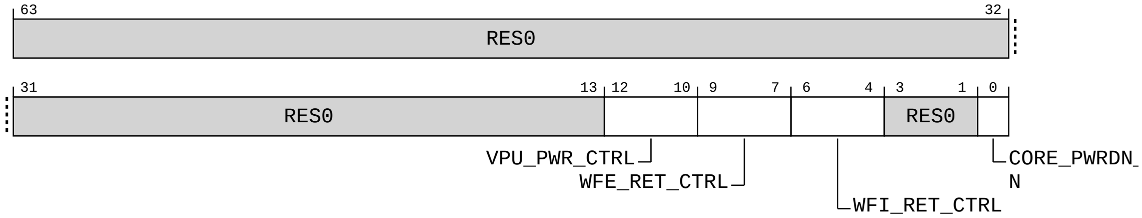


Table B-46: IMP_CPUPWRCTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:13]	RES0	Reserved	0b0
[12:10]	VPU_PWR_CTRL	VPU power down control. **000** VPU powerdown is disabled. **001** 2 system counter ticks are required before VPU powerdown. **010** 8 system counter ticks are required before VPU powerdown. **011** 32 system counter ticks are required before VPU powerdown. **100** 64 system counter ticks are required before VPU powerdown. **101** 128 system counter ticks are required before VPU powerdown. **110** 256 system counter ticks are required before VPU powerdown. **111** 512 system counter ticks are required before VPU powerdown.	

Bits	Name	Description	Reset
[9:7]	WFE_RET_CTRL	Wait for Event retention control. **`000`** Dynamic retention is disabled. **`001`** 2 system counter ticks are required before retention entry. **`010`** 8 system counter ticks are required before retention entry. **`011`** 32 system counter ticks are required before retention entry. **`100`** 64 system counter ticks are required before retention entry. **`101`** 128 system counter ticks are required before retention entry. **`110`** 256 system counter ticks are required before retention entry. **`111`** 512 system counter ticks are required before retention entry.	
[6:4]	WFI_RET_CTRL	Wait for Interrupt retention control. **`000`** Dynamic retention is disabled. **`001`** 2 system counter ticks are required before retention entry. **`010`** 8 system counter ticks are required before retention entry. **`011`** 32 system counter ticks are required before retention entry. **`100`** 64 system counter ticks are required before retention entry. **`101`** 128 system counter ticks are required before retention entry. **`110`** 256 system counter ticks are required before retention entry. **`111`** 512 system counter ticks are required before retention entry.	
[3:1]	RES0	Reserved	`0b0`
[0]	CORE_PWRDN_EN	Indicates to the power controller if the CPU wants to power down when it enters WFE/WFI state.	

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;
elsif 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];
```

B.1.16 IMP_ATCR_EL2, CPU Auxiliary Translation Control Register

This register controls the values of the PBHA signals for memory accesses generated by translation table walks in the EL2 translation regime.

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 B-16: AArch64_imp_atcr_el2 bit assignments

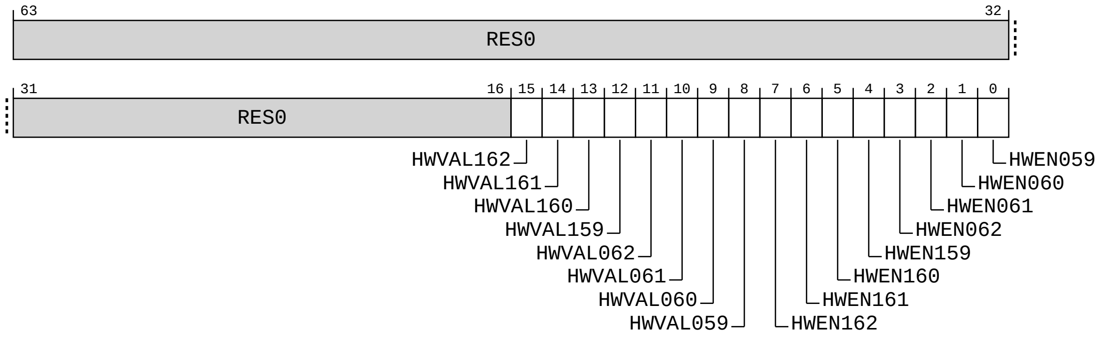


Table B-49: IMP_ATCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0b0
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to page table walks using TTBR1_EL2 if HWEN162 is set.	
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to page table walks using TTBR1_EL2 if HWEN161 is set.	
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to page table walks using TTBR1_EL2 if HWEN160 is set.	
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to page table walks using TTBR1_EL2 if HWEN159 is set.	
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to page table walks using TTBR0_EL2 if HWEN062 is set.	
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to page table walks using TTBR0_EL2 if HWEN061 is set.	
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to page table walks using TTBR0_EL2 if HWEN060 is set.	
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to page table walks using TTBR0_EL2 if HWEN059 is set.	
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR1_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR1_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR1_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR1_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	

Bits	Name	Description	Reset
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR0_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	

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;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    return IMP_ATCR_EL2;
elsif PSTATE.EL == EL3 then
    return IMP_ATCR_EL2;
```

MSR S3_4_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 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];
```

B.1.17 IMP_AVTCR_EL2, CPU Auxiliary Translation Control Register

This register controls the values of the PBHA signals for memory accesses generated by stage 2 translation table walks.

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 B-17: AArch64_imp_avtcr_el2 bit assignments

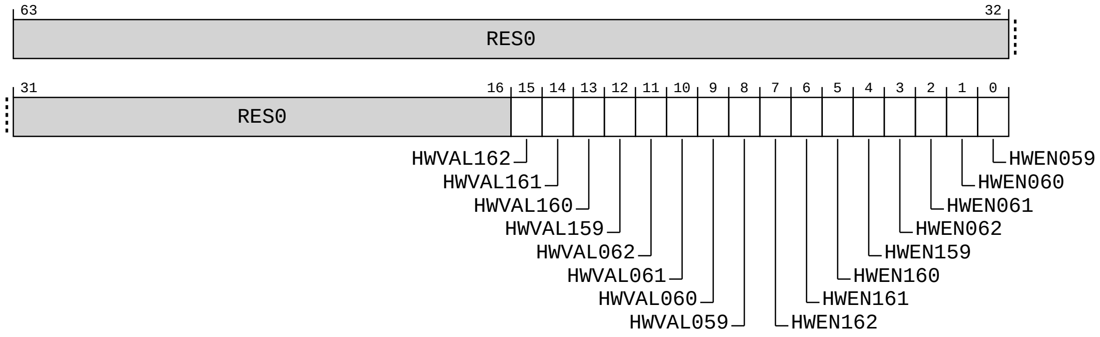


Table B-52: IMP_AVTCR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0b0
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN162 is set.	
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN161 is set.	
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN160 is set.	
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to page table walks using VSTTBR_EL2 if HWEN159 is set.	
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to page table walks using VTTBR_EL2 if HWEN062 is set.	
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to page table walks using VTTBR_EL2 if HWEN061 is set.	
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to page table walks using VTTBR_EL2 if HWEN060 is set.	
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to page table walks using VTTBR_EL2 if HWEN059 is set.	
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to page table walks using VSTTBR_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using VTTBR_EL2. If this bit is clear, PBHA[3] will be 0 on page table walks.	

Bits	Name	Description	Reset
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using VTTBR_EL2. If this bit is clear, PBHA[2] will be 0 on page table walks.	
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using VTTBR_EL2. If this bit is clear, PBHA[1] will be 0 on page table walks.	
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using VTTBR_EL2. If this bit is clear, PBHA[0] will be 0 on page table walks.	

Access

MRS <Xt>, 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, <Xt>

<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.TIDCP == 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.TIDCP == 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];
```

B.1.18 IMP_ATCR_EL1, CPU Auxiliary Translation Control Register

This register controls the values of the PBHA signals for memory accesses generated by translation table walks in the EL1 translation regime.

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 B-18: AArch64_imp_atcr_el1 bit assignments

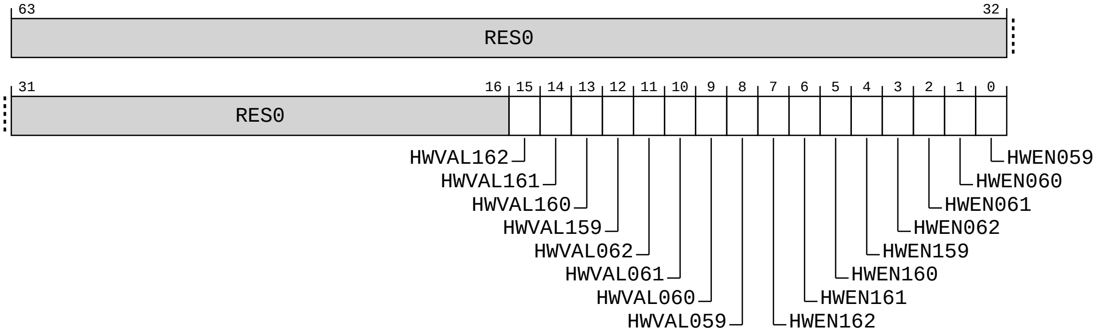


Table B-55: IMP_ATCR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	`0b0`
[15]	HWVAL162	Value of PBHA[3] on memory accesses due to page table walks using TTBR1_EL1 if HWEN162 is set.	
[14]	HWVAL161	Value of PBHA[2] on memory accesses due to page table walks using TTBR1_EL1 if HWEN161 is set.	
[13]	HWVAL160	Value of PBHA[1] on memory accesses due to page table walks using TTBR1_EL1 if HWEN160 is set.	
[12]	HWVAL159	Value of PBHA[0] on memory accesses due to page table walks using TTBR1_EL1 if HWEN159 is set.	
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to page table walks using TTBR0_EL1 if HWEN062 is set.	
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to page table walks using TTBR0_EL1 if HWEN061 is set.	
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to page table walks using TTBR0_EL1 if HWEN060 is set.	
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to page table walks using TTBR0_EL1 if HWEN059 is set.	
[7]	HWEN162	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR1_EL1. If this bit is clear, PBHA[3] will be 0 on page table walks.	

Bits	Name	Description	Reset
[6]	HWEN161	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR1_EL1. If this bit is clear, PBHA[2] will be 0 on page table walks.	
[5]	HWEN160	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR1_EL1. If this bit is clear, PBHA[1] will be 0 on page table walks.	
[4]	HWEN159	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR1_EL1. If this bit is clear, PBHA[0] will be 0 on page table walks.	
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR0_EL1. If this bit is clear, PBHA[3] will be 0 on page table walks.	
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR0_EL1. If this bit is clear, PBHA[2] will be 0 on page table walks.	
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR0_EL1. If this bit is clear, PBHA[1] will be 0 on page table walks.	
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR0_EL1. If this bit is clear, PBHA[0] will be 0 on page table walks.	

Access

MRS <Xt>, S3_0_C15_C7_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C7_0	0b11	0b000	0b1111	0b0111	0b000

MSR S3_0_C15_C7_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C7_0	0b11	0b000	0b1111	0b0111	0b000

MRS <Xt>, S3_5_C15_C7_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_5_C15_C7_0	0b11	0b101	0b1111	0b0111	0b000

MSR S3_5_C15_C7_0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S3_5_C15_C7_0	0b11	0b101	0b1111	0b0111	0b000

Accessibility

MRS <Xt>, S3_0_C15_C7_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_ATCR_EL1;
elsif PSTATE.EL == EL2 then
    return IMP_ATCR_EL1;
elsif PSTATE.EL == EL3 then
```

```
    return IMP_ATCR_EL1;
```

MSR S3_0_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        IMP_ATCR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    IMP_ATCR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL1 = X[t];
```

MRS <Xt>, S3_5_C15_C7_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if EL2Enabled() && HCR_EL2.E2H == 1 then
        return IMP_ATCR_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == 1 then
        return IMP_ATCR_EL1;
    else
        UNDEFINED;
```

MSR S3_5_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if EL2Enabled() && HCR_EL2.E2H == 1 then
        IMP_ATCR_EL1 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == 1 then
        IMP_ATCR_EL1 = X[t];
    else
        UNDEFINED;
```

B.1.19 IMP_ATCR_EL3, CPU Auxiliary Translation Control Register

This register controls the values of the PBHA signals for memory accesses generated by translation table walks in the EL3 translation regime.

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 B-19: AArch64_imp_atcr_el3 bit assignments

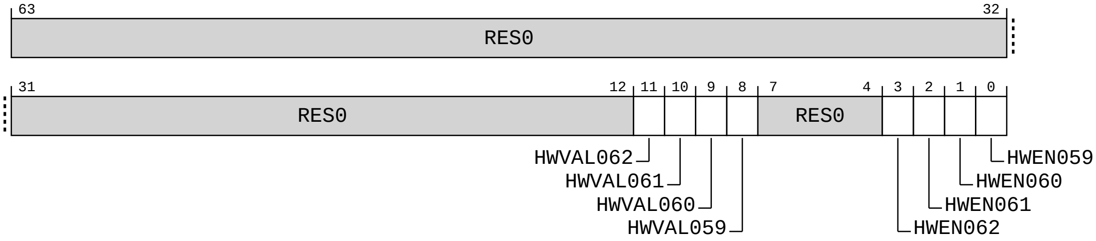


Table B-60: IMP_ATCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:12]	RES0	Reserved	0b0
[11]	HWVAL062	Value of PBHA[3] on memory accesses due to page table walks using TTBR0_EL3 if HWEN062 is set.	
[10]	HWVAL061	Value of PBHA[2] on memory accesses due to page table walks using TTBR0_EL3 if HWEN061 is set.	
[9]	HWVAL060	Value of PBHA[1] on memory accesses due to page table walks using TTBR0_EL3 if HWEN060 is set.	
[8]	HWVAL059	Value of PBHA[0] on memory accesses due to page table walks using TTBR0_EL3 if HWEN059 is set.	
[7:4]	RES0	Reserved	0b0
[3]	HWEN062	Enable use of PBHA[3] on memory accesses due to page table walks using TTBR0_EL3. If this bit is clear, PBHA[3] will be 0 on page table walks.	
[2]	HWEN061	Enable use of PBHA[2] on memory accesses due to page table walks using TTBR0_EL3. If this bit is clear, PBHA[2] will be 0 on page table walks.	
[1]	HWEN060	Enable use of PBHA[1] on memory accesses due to page table walks using TTBR0_EL3. If this bit is clear, PBHA[1] will be 0 on page table walks.	
[0]	HWEN059	Enable use of PBHA[0] on memory accesses due to page table walks using TTBR0_EL3. If this bit is clear, PBHA[0] will be 0 on page table walks.	

Access

MRS <Xt>, 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, <Xt>

<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;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_ATCR_EL3;
```

MSR S3_6_C15_C7_0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_ATCR_EL3 = X[t];
```

B.1.20 AFSR0_EL2, Auxiliary Fault Status Register 0 (EL2)

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

0x0

Bit descriptions

Figure B-20: AArch64_afsr0_el2 bit assignments

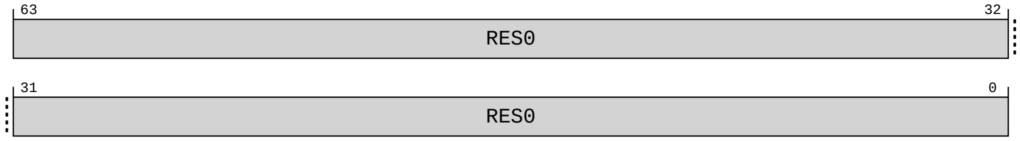


Table B-63: AFSR0_EL2 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

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
    UNDEFINED;
```

```
elsif PSTATE.EL == EL2 then
    return AFSR0_EL2;
elsif PSTATE.EL == EL3 then
    return AFSR0_EL2;
```

MSR AFSR0_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    AFSR0_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR0_EL2 = X[t];
```

MRS <Xt>, AFSR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return AFSR0_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == 1 then
        return AFSR0_EL2;
    else
        return AFSR0_EL1;
elsif PSTATE.EL == EL3 then
    return AFSR0_EL1;
```

MSR AFSR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == 1 then
        AFSR0_EL2 = X[t];
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t];
```

B.1.21 AFSR1_EL2, Auxiliary Fault Status Register 1 (EL2)

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

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Generic system control

Reset value

`0x0`

Bit descriptions

Figure B-21: AArch64_afsr1_el2 bit assignments

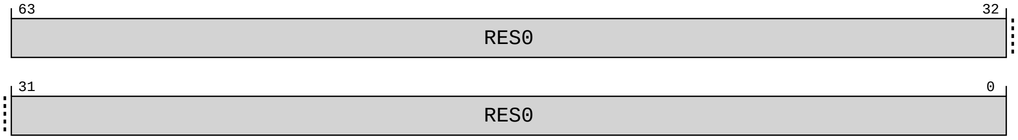


Table B-68: AFSR1_EL2 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

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;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    return AFSR1_EL2;
elsif PSTATE.EL == EL3 then
    return AFSR1_EL2;
```

MSR AFSR1_EL2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    AFSR1_EL2 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR1_EL2 = X[t];
```

MRS <Xt>, AFSR1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return AFSR1_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == 1 then
        return AFSR1_EL2;
    else
        return AFSR1_EL1;
elsif 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);
    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];
```

B.1.22 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 B-22: AArch64_afsr0_el1 bit assignments

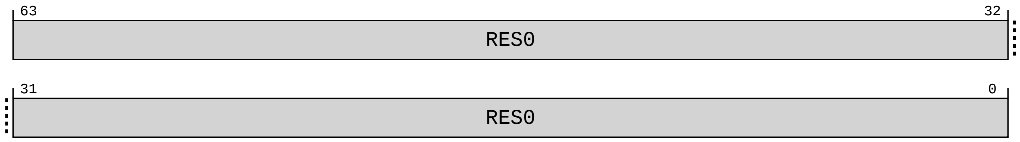


Table B-73: AFSR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	`0b0`

Access

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`

MRS <Xt>, AFSR0_EL12

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL12	`0b11`	`0b101`	`0b0101`	`0b0001`	`0b000`

MSR AFSR0_EL12, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL12	0b11	0b101	0b0101	0b0001	0b000

Accessibility

MRS <Xt>, AFSR0_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return AFSR0_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == 1 then
        return AFSR0_EL2;
    else
        return AFSR0_EL1;
elsif PSTATE.EL == EL3 then
    return AFSR0_EL1;
```

MSR AFSR0_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TVM == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == 1 then
        AFSR0_EL2 = X[t];
    else
        AFSR0_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    AFSR0_EL1 = X[t];
```

MRS <Xt>, AFSR0_EL12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == 1 then
        return AFSR0_EL1;
    else
        UNDEFINED;
elsif 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;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == 1 then
        AFSR0_EL1 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == 1 then
        AFSR0_EL1 = X[t];
    else
        UNDEFINED;
```

B.1.23 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 B-23: AArch64_afsr1_el1 bit assignments

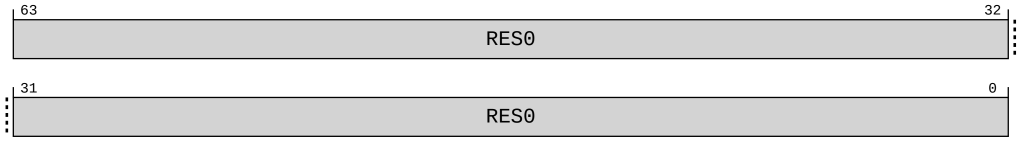


Table B-78: AFSR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

Access

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

MRS <Xt>, AFSR1_EL12

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL12	0b11	0b101	0b0101	0b0001	0b001

MSR AFSR1_EL12, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR1_EL12	0b11	0b101	0b0101	0b0001	0b001

Accessibility

MRS <Xt>, AFSR1_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TRVM == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return AFSR1_EL1;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == 1 then
        return AFSR1_EL2;
    else
        return AFSR1_EL1;
elsif 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);
    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];
```

MRS <Xt>, AFSR1_EL12

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == 1 then
        return AFSR1_EL1;
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == 1 then
        return AFSR1_EL1;
    else
        UNDEFINED;
```

MSR AFSR1_EL12, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == 1 then
        AFSR1_EL1 = X[t];
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && HCR_EL2.E2H == 1 then
        AFSR1_EL1 = X[t];
    else
        UNDEFINED;
```

B.1.24 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 B-24: AArch64_afsr0_el3 bit assignments

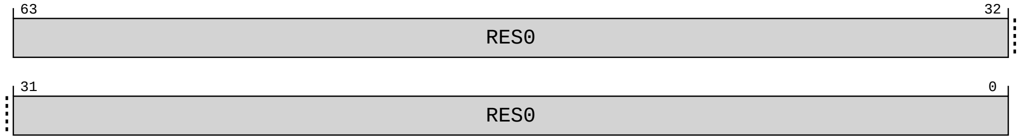


Table B-83: AFSR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

Access

MRS <Xt>, AFSR0_EL3

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL3	0b11	0b110	0b0101	0b0001	0b000

MSR AFSR0_EL3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
AFSR0_EL3	0b11	0b110	0b0101	0b0001	0b000

Accessibility

MRS <Xt>, AFSR0_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 AFSR0_EL3;
```

MSR AFSR0_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
    AFSR0_EL3 = X[t];
```

B.1.25 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 B-25: AArch64_afsr1_el3 bit assignments

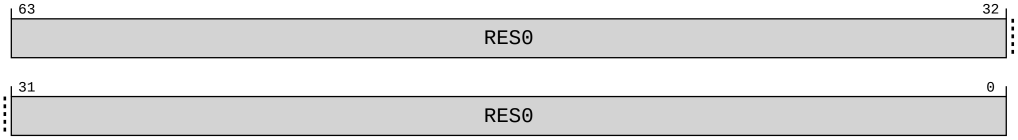


Table B-86: AFSR1_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

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;
elsif PSTATE.EL == EL1 then
```

```
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return AFSR1_EL3;
```

MSR AFSR1_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
    AFSR1_EL3 = X[t];
```

B.2 Special purpose register summary

The summary table provides an overview of special purpose registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table B-89: special-purpose register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
IMP_CPUPPMCR_EL3	3	C15	6	C2	0	See individual bit resets.	64-bit	Global PPM Configuration Register

B.2.1 IMP_CPUPPMCR_EL3, Global PPM Configuration Register

This register controls global PPM features and allows discovery of some PPM implementation details.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

special-purpose

Reset value

See individual bit resets.

Bit descriptions

Figure B-26: AArch64_imp_cpuppmcr_el3 bit assignments

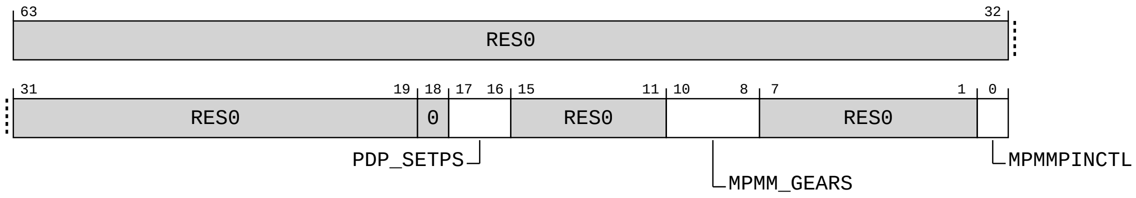


Table B-90: IMP_CPUPPMCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:18]	RES0	Reserved	0b0
[17:16]	PDP_SETPS	Number of PDP Setpoints implemented **00** PDP is not implemented or enabled.	
[15:11]	RES0	Reserved	0b0
[10:8]	MPMM_GEARS	Number of MPMM Gears implemented **011** 3 MPMM are enabled.	
[7:1]	RES0	Reserved	0b0
[0]	MPMMPINCTL	MPMM Pin Control Enabled **0** MPMM control through SPR and utility bus. **1** MPMM control through pin only.	0b0

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
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
```

```
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        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
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    IMP_CPUPPMCR_EL3 = X[t];
```

B.3 Debug register summary

The summary table provides an overview of debug registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table B-93: Debug register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
IMP_CDBGDR0_EL3	3	C15	6	C0	0	See individual bit resets.	64-bit	Cache Debug Data Register 0

B.3.1 IMP_CDBGDR0_EL3, Cache Debug Data Register 0

Contains data from a preceding cache debug operation.

This register is populated after one of the following operations have been executed:

- SYS IMP_CDBGL1DCDR

- SYS IMP_CDBGL1DCMR

- SYS IMP_CDBGL1DCTR

- SYS IMP_CDBGL1ICDR

- SYS IMP_CDBGL1ICTR

- SYS IMP_CDBGL2CDR

- SYS IMP_CDBGL2CMR

- SYS IMP_CDBGL2CTR
- SYS IMP_CDBGL2TR0
- SYS IMP_CDBGL2TR1
- SYS IMP_CDBGL2TR2

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

debug

Reset value

See individual bit resets.

Bit descriptions

After SYS IMP_CDBGL1DCDR or SYS IMP_CDBGL2CDR operations

Figure B-27: AArch64_imp_cdbgdr0_el3 bit assignments

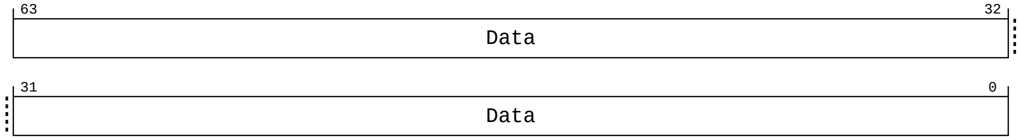


Table B-94: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:0]	Data	Data contents of cache at specified Set/Way/Offset	

After SYS IMP_CDBGL1DCMR or SYS IMP_CDBGL2CMR operations

Figure B-28: AArch64_imp_cdbgdr0_el3 bit assignments

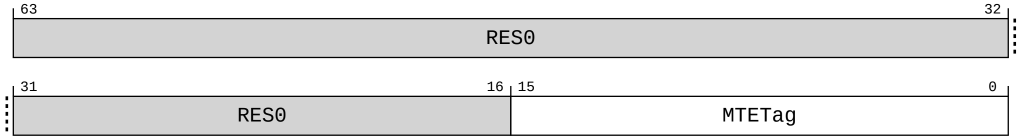


Table B-95: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	`0b0`
[15:0]	MTETag	MTE tag contents of cache at specified Set/Way	

After a SYS IMP_CDBGL1ICDR operation

Figure B-29: AArch64_imp_cdbgdr0_el3 bit assignments

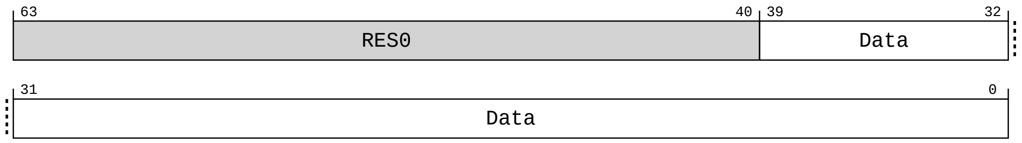


Table B-96: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	`0b0`
[39:0]	Data	Data contents of cache at specified Set/Way/Offset	

After a SYS IMP_CDBGL1DCTR operation

Figure B-30: AArch64_imp_cdbgdr0_el3 bit assignments

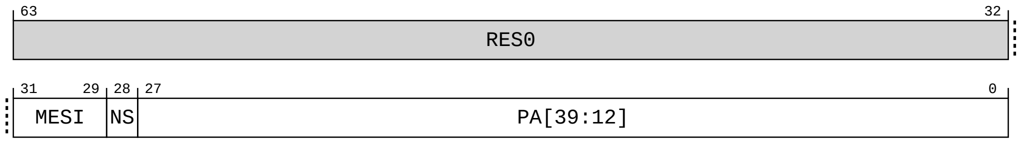


Table B-97: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0b0`
[31:29]	MESI	Partial MESI state **`000`** Invalid **`001`** Shared **`010`** Unique Non-transient **`011`** Unique Transient	

Bits	Name	Description	Reset
[28]	NS	Tag security state **0** Secure **1** Non-secure	
[27:0]	PA[39:12]	Tag physical address	

After a SYS IMP_CDBGL1DCDTR operation

Figure B-31: AArch64_imp_cdbgdr0_el3 bit assignments

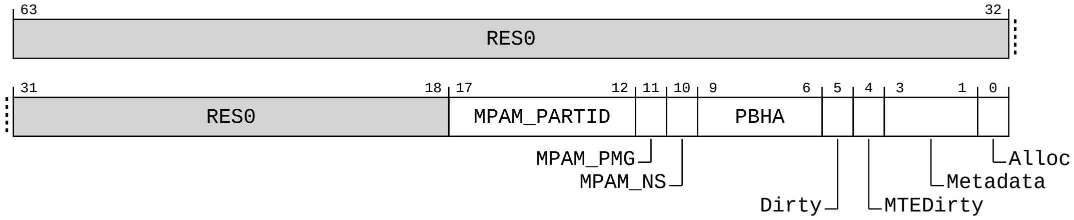


Table B-98: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:18]	RES0	Reserved	0b0
[17:12]	MPAM_PARTID	MPAM partition ID	
[11]	MPAM_PMG	MPAM performance monitoring group	
[10]	MPAM_NS	Indicates MPAM PARTID space **0** Secure physical PARTID space **1** Non-secure physical PARTID space	
[9:6]	PBHA	Page-Based Hardware Attributes	
[5]	Dirty	Indicates whether the cache line data is dirty	
[4]	MTEDirty	Indicates whether the MTE tag data for the cache line is dirty	
[3:1]	Metadata	Internal metadata	
[0]	Alloc	Outer allocation hint **0** No write allocate **1** Write allocate	

After a SYS IMP_CDBGL1ICTR operation

Figure B-32: AArch64_imp_cdbgdr0_el3 bit assignments

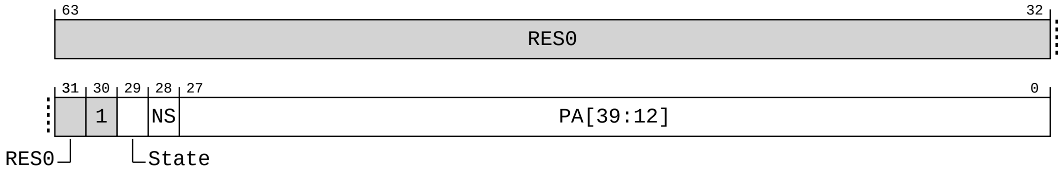


Table B-99: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	0b0
[30]	RES1	Reserved	0b1
[29]	State	Cache line state **0** Valid **1** Invalid	
[28]	NS	Tag security state **0** Secure **1** Non-secure	
[27:0]	PA[39:12]	Tag physical address	

After a IMP_CDBGL2CTR operation

Figure B-33: AArch64_imp_cdbgdr0_el3 bit assignments

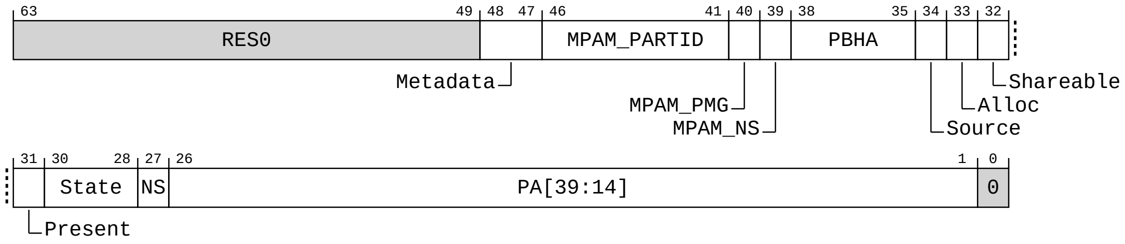


Table B-100: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:49]	RES0	Reserved	0b0
[48:47]	Metadata	Internal metadata	
[46:41]	MPAM_PARTID	MPAM partition ID	
[40]	MPAM_PMG	MPAM performance monitoring group	

Bits	Name	Description	Reset

[39]	MPAM_NS	Indicates MPAM PARTID space
		`0`
		Secure physical PARTID space
		`1`
		Non-secure physical PARTID space	

[38:35]	PBHA	Page-Based Hardware Attributes	

[34]	Source	Cache line source
		`0`
		Line was brought into complex from outside the cluster
		`1`
		Line was brought into complex from an L3 hit	

[33]	Alloc	Outer allocation hint
		`0`
		No write allocate
		`1`
		Write allocate	

[32]	Shareable	Cache line shareability
		`0`
		Non-shareable
		`1`
		Outer Shareable	

[31]	Present	Cache line is present in the L1 cache of any of the cores in this complex.	

[30:28]	State	Cache line state
		`000`
		Invalid
		`001`
		SharedClean, MTE tags invalid
		`010`
		UniqueClean, MTE tags invalid
		`011`
		UniqueDirty, MTE tags invalid
		`100`
		SharedClean, MTE tags clean
		`101`
		UniqueClean, MTE tags clean
		`110`
		UniqueDirty, MTE tags clean
		`111`
		UniqueDirty, MTE tags dirty	

Bits	Name	Description	Reset
[27]	NS	Tag security state **0** Secure **1** Non-secure	
[26:1]	PA[39:14]	Tag physical address	
[0]	RES0	Reserved	`0b0`

After a SYS IMP_CDBGL2TR0 operation

Figure B-34: AArch64_imp_cdbgdr0_el3 bit assignments

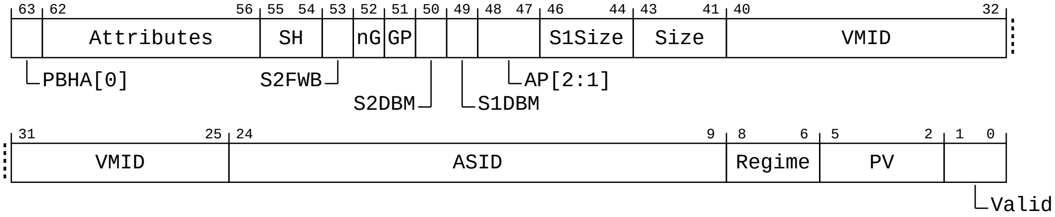


Table B-101: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63]	PBHA[0]	Lower bit of Page-Based Hardware Attributes	
[62:56]	Attributes	Memory attributes for the entry **x000x00** Device-nGnRnE. **x000x01** Device-nGnRE. **x000x10** Device-nGRE. **x000x11** Device-GRE.	

Bits	Name	Description	Reset
[62:56]	Attributes	**x0010:Outer[1:0]** Normal memory, Inner Non-cacheable, Outer Write-Back. Outer[1:0] are the outer allocation hints. **x0011:Outer[1:0]** Normal memory, Inner Write-Through, Outer Write-Back. Outer[1:0] are the outer allocation hints. **x0100:Outer[1:0]** Normal memory, Inner Non-cacheable, Outer Write-Through. Outer[1:0] are the outer allocation hints. **x0101:Outer[1:0]** Normal memory, Inner Write-Back, Outer Write-Through. Outer[1:0] are the outer allocation hints. **x0110:Outer[1:0]** Normal memory, Inner Write-Through, Outer Write-Through. Outer[1:0] are the outer allocation hints. **x011100** Normal memory, Inner Non-cacheable, Outer Non-cacheable. **x011101** Normal memory, Inner Write-Back, Outer Non-cacheable. **x011110** Normal memory, Inner Write-Through, Outer Non-cacheable. **010:Inner[1:0]:Outer[1:0]** Normal memory, Inner Write-Back, Outer Write-Back Non-transient. Inner[1:0] are the inner allocation hints and Outer[1:0] are the outer allocation hints. **011:Inner[1:0]:Outer[1:0]** Normal memory, Inner Write-Back, Outer Write-Back Transient. Inner[1:0] are the inner allocation hints and Outer[1:0] are the outer allocation hints. **110:Inner[1:0]:Outer[1:0]** Tagged Normal memory, Inner Write-Back, Outer Write-Back Non-transient. Inner[1:0] are the inner allocation hints and Outer[1:0] are the outer allocation hints.	
[55:54]	SH	Shareability **00** Non-shareable **10** Outer Shareable **11** Inner Shareable **Note:** Device memory is always outer shareable.	
[53]	S2FWB	Stage 2 forced attributes to be WB	
[52]	nG	Not global	
[51]	GP	Guarded page	
[50]	S2DBM	Stage 2 Dirty Bit Modifier	
[49]	S1DBM	Stage 1 Dirty Bit Modifier	
[48:47]	AP[2:1]	Stage 1 access permissions	

Bits	Name	Description	Reset

[46:44]	S1Size	The original size of the stage 1 translation
		`000`
		4KB or 16KB
		`001`
		64KB
		`010`
		2MB
		`011`
		8MB
		`100`
		32MB
		`101`
		128MB
		`110`
		512MB
		`111`
		1GB	

[43:41]	Size	The size of the entry
		`000`
		16KB
		`001`
		64KB
		`010`
		2MB
		`011`
		8MB
		`100`
		32MB
		`101`
		128MB
		`110`
		512MB
		`111`
		1GB	

[40:25]	VMID	VMID value, when supported by the regime	
[24:9]	ASID	ASID value, when supported by the regime	

Bits	Name	Description	Reset

[8:6]	Regime	Translation regime used to fetch the entry
		`000`
		Secure EL1&0
		`001`
		Secure EL2&0
		`010`
		Secure EL2
		`011`
		Secure EL3
		`100`
		Non-secure EL1&0
		`101`
		Non-secure EL2&0
		`110`
		Non-secure EL2	

After a SYS IMP_CDBGL2TR1 operation

Figure B-35: AArch64_imp_cdbgdr0_el3 bit assignments

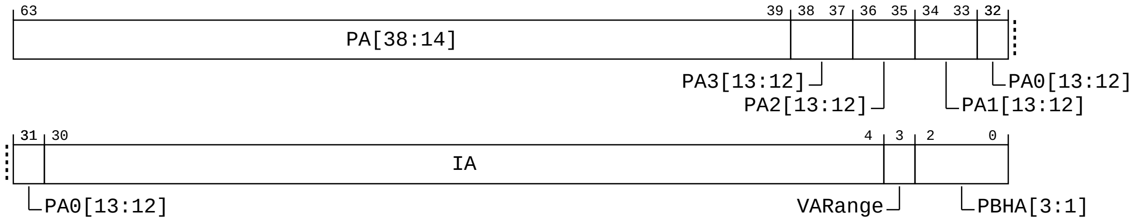


Table B-102: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:39]	PA[38:14]	The lower bits [38:14] of the PA	
[38:37]	PA3[13:12]	Low bits of PA for the clustered entry with VA[13:12] == 3 (only for 16k pages)	
[36:35]	PA2[13:12]	Low bits of PA for the clustered entry with VA[13:12] == 2 (only for 16k pages)	
[34:33]	PA1[13:12]	Low bits of PA for the clustered entry with VA[13:12] == 1 (only for 16k pages)	
[32:31]	PA0[13:12]	Low bits of PA for the clustered entry with VA[13:12] == 0 (only for 16k pages)	
[30:4]	IA	Intermediate Address For stage 1 translations, this field is VA[47:21]. For stage 2 translations, this field is {NS, IPA[39:21], `7’h00`}.	

Bits	Name	Description	Reset

[3]	VARange	The VA range for translation regimes which support two VA ranges
		`0`
		Lower VA range
		`1`
		Upper VA range	

[2:0]	PBHA[3:1]	Upper bits of Page-Based Hardware Attributes	

After a SYS IMP_CDBGL2TR2 operation

Figure B-36: AArch64_imp_cdbgdr0_el3 bit assignments

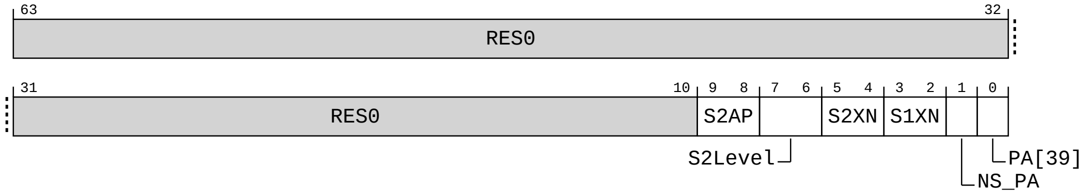


Table B-103: IMP_CDBGDR0_EL3 bit descriptions

Bits	Name	Description	Reset
[63:10]	RES0	Reserved	`0b0`
[9:8]	S2AP	Stage 2 access permissions	
[7:6]	S2Level	Final level of stage 2 page walk used to generate PA	
[5:4]	S2XN	S2 execute never permissions	
[3:2]	S1XN	S1 execute never permissions	
[1]	NS_PA	NS bit for the PA space	
[0]	PA[39]	Bit [39] of the PA	

Access

MRS <Xt>, S3_6_C15_C0_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_6_C15_C0_0	`0b11`	`0b110`	`0b1111`	`0b0000`	`0b000`

Accessibility

MRS <Xt>, S3_6_C15_C0_0

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
```

```
elsif PSTATE.EL == EL3 then
    return IMP_CDBGDR0_EL3;
```

B.4 System instruction register summary

The summary table provides an overview of System instruction registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table B-105: System instruction register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
SYS_IMP_CDBGL1DCTR	1	C15	6	C2	0	See individual bit resets.	64-bit	L1 Data Cache Tag Read Operation
SYS_IMP_CDBGL1ICTR	1	C15	6	C2	1	See individual bit resets.	64-bit	L1 Instruction Cache Tag Read Operation
SYS_IMP_CDBGL2TR0	1	C15	6	C2	2	See individual bit resets.	64-bit	L2 TLB Read Operation 0
SYS_IMP_CDBGL2CTR	1	C15	6	C2	3	See individual bit resets.	64-bit	L2 Cache Tag Read Operation
SYS_IMP_CDBGL1DCDTR	1	C15	6	C2	4	See individual bit resets.	64-bit	L1 Data Cache Dirty Read Operation
SYS_IMP_CDBGL1DCMR	1	C15	6	C3	0	See individual bit resets.	64-bit	L1 Data Cache MTE Tag Read Operation
SYS_IMP_CDBGL2TR1	1	C15	6	C3	2	See individual bit resets.	64-bit	L2 TLB Read Operation 1
SYS_IMP_CDBGL2CMR	1	C15	6	C3	3	See individual bit resets.	64-bit	L2 Cache MTE Tag Read Operation
SYS_IMP_CDBGL1DCDR	1	C15	6	C4	0	See individual bit resets.	64-bit	L1 Data Cache Data Read Operation
SYS_IMP_CDBGL1ICDR	1	C15	6	C4	1	See individual bit resets.	64-bit	L1 Instruction Cache Data Read Operation
SYS_IMP_CDBGL2TR2	1	C15	6	C4	2	See individual bit resets.	64-bit	L2 TLB Read Operation 2
SYS_IMP_CDBGL2CDR	1	C15	6	C4	3	See individual bit resets.	64-bit	L2 Cache Data Read Operation

B.4.1 SYS IMP_CDBGL1DCTR, L1 Data Cache Tag Read Operation

Read contents of the L1 Data Cache Tag Memory.

The cache tag is written to AArch64-IMP_CDBGDR0_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 B-37: AArch64_sys_imp_cdbgl1dctr bit assignments

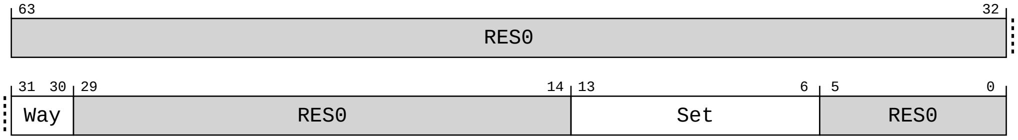


Table B-106: SYS IMP_CDBGL1DCTR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:30]	Way	Cache way	
[29:14]	RES0	Reserved	0b0
[13:6]	Set	Cache set	
[5:0]	RES0	Reserved	0b0

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C2, #0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C2_0	0b01	0b110	0b1111	0b0010	0b000

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C2, #0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL1DCTR(X[t]);
```

B.4.2 SYS IMP_CDBGL1ICTR, L1 Instruction Cache Tag Read Operation

Read contents of the L1 Instruction Cache Tag Memory.

The cache tag is written to AArch64-IMP_CDBGDR0_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 B-38: AArch64_sys_imp_cdbgl1ictr bit assignments

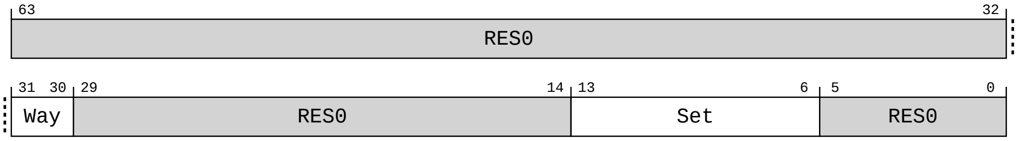


Table B-108: SYS IMP_CDBGL1ICTR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:30]	Way	Cache way	
[29:14]	RES0	Reserved	0b0
[13:6]	Set	Cache set	
[5:0]	RES0	Reserved	0b0

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C2, #1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C2_1	0b01	0b110	0b1111	0b0010	0b001

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C2, #1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
```

```
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL1ICTR(X[t]);
```

B.4.3 SYS IMP_CDBGL2TR0, L2 TLB Read Operation 0

Read contents of the Level 2 TLB Memory.

Bits [63:0] of the TLB data are written to AArch64-IMP_CDBGDR0_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 B-39: AArch64_sys_imp_cdbgl2tr0 bit assignments

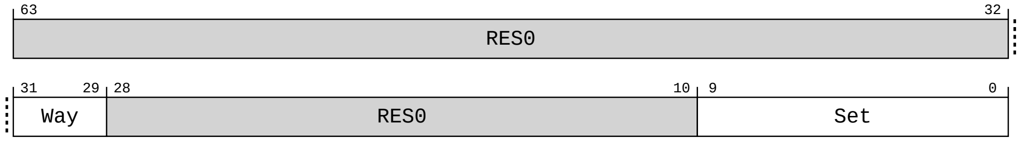


Table B-110: SYS IMP_CDBGL2TR0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:29]	Way	TLB way	
[28:10]	RES0	Reserved	0b0
[9:0]	Set	TLB set. For a single-CPU configuration sets 0x000-0x07F access the main TLB and sets 0x080-0x0A4 access the TLB for IPA and walk entries. For a dual-core configuration sets 0x000-0x0FF access the main TLB and sets 0x100-0x147 access the TLB for IPA and walk entries.	

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C2, #2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C2_2	0b01	0b110	0b1111	0b0010	0b010

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C2, #2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL2TR0(X[t]);
```

B.4.4 SYS IMP_CDBGL2CTR, L2 Cache Tag Read Operation

Read contents of the L2 Cache Tag Memory.

The cache tag is written to AArch64-IMP_CDBGDR0_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 B-40: AArch64_sys_imp_cdbgl2ctr bit assignments

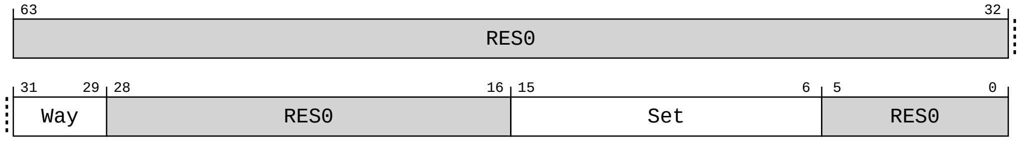


Table B-112: SYS IMP_CDBGL2CTR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0b0`
[31:29]	Way	Cache way	
[28:16]	RES0	Reserved	`0b0`
[15:6]	Set	Cache set	
[5:0]	RES0	Reserved	`0b0`

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C2, #3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C2_3	`0b01`	`0b110`	`0b1111`	`0b0010`	`0b011`

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C2, #3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL2CTR(X[t]);
```

B.4.5 SYS IMP_CDBGL1DCDTR, L1 Data Cache Dirty Read Operation

Read contents of the L1 Data Cache Dirty Memory.

The cache tag is written to AArch64-IMP_CDBGDR0_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 B-41: AArch64_sys_imp_cdbgl1dcdtr bit assignments

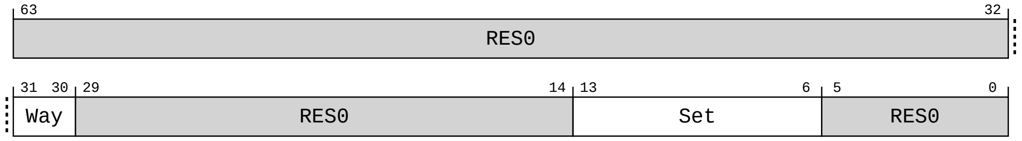


Table B-114: SYS IMP_CDBGL1DCDTR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:30]	Way	Cache way	
[29:14]	RES0	Reserved	0b0
[13:6]	Set	Cache set	
[5:0]	RES0	Reserved	0b0

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C2, #4, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C2_4	0b01	0b110	0b1111	0b0010	0b100

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C2, #4, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL1DCDTR(X[t]);
```

B.4.6 SYS IMP_CDBGL1DCMR, L1 Data Cache MTE Tag Read Operation

Read contents of the L1 Data Cache MTE Tag Memory.

The 16 bits of cache MTE tag data are written to AArch64-IMP_CDBGDR0_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 B-42: AArch64_sys_imp_cdbgl1dcmr bit assignments

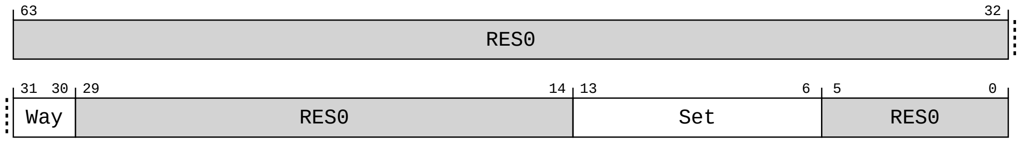


Table B-116: SYS IMP_CDBGL1DCMR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:30]	Way	Cache way	
[29:14]	RES0	Reserved	0b0
[13:6]	Set	Cache set	
[5:0]	RES0	Reserved	0b0

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C3, #0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C3_0	0b01	0b110	0b1111	0b0011	0b000

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C3, #0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL1DCMR(X[t]);
```

B.4.7 SYS IMP_CDBGL2TR1, L2 TLB Read Operation 1

Read contents of the Level 2 TLB Memory.

Bits [127:64] of the TLB data are written to AArch64-IMP_CDBGDR0_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 B-43: AArch64_sys_imp_cdbgl2tr1 bit assignments

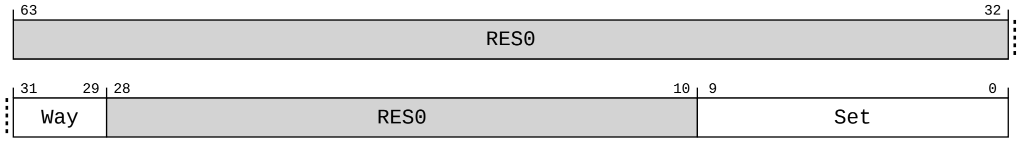


Table B-118: SYS IMP_CDBGL2TR1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:29]	Way	TLB way	
[28:10]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[9:0]	Set	TLB set. For a single-CPU configuration sets 0x000-0x07F access the main TLB and sets 0x080-0x0A4 access the TLB for IPA and walk entries. For a dual-core configuration sets 0x000-0x0FF access the main TLB and sets 0x100-0x147 access the TLB for IPA and walk entries.	

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C3, #2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C3_2	0b01	0b110	0b1111	0b0011	0b010

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C3, #2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL2TR1(X[t]);
```

B.4.8 SYS IMP_CDBGL2CMR, L2 Cache MTE Tag Read Operation

Read contents of the L2 Cache MTE Tag Memory.

The 16 bits of cache MTE tag data are written to AArch64-IMP_CDBGDR0_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 B-44: AArch64_sys_imp_cdbgl2cmr bit assignments

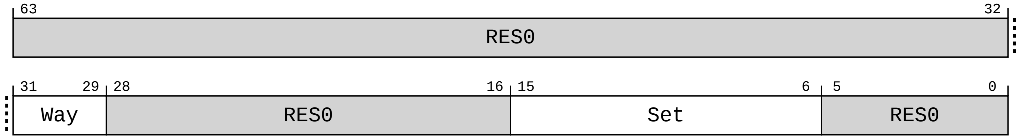


Table B-120: SYS IMP_CDBGL2CMR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:29]	Way	Cache way	
[28:16]	RES0	Reserved	0b0
[15:6]	Set	Cache set	
[5:0]	RES0	Reserved	0b0

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C3, #3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C3_3	0b01	0b110	0b1111	0b0011	0b011

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C3, #3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL2CMR(X[t]);
```

B.4.9 SYS IMP_CDBGL1DCDR, L1 Data Cache Data Read Operation

Read contents of the L1 Data Cache Data Memory.

The 64 bits of cache data are written to AArch64-IMP_CDBGDR0_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 B-45: AArch64_sys_imp_cdbgl1dcdr bit assignments

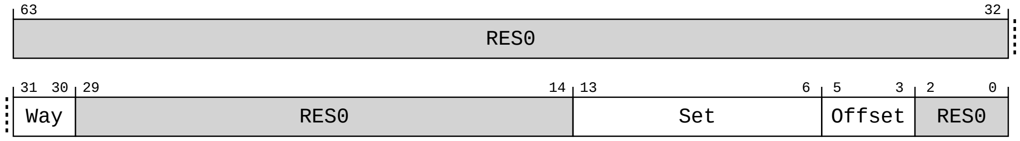


Table B-122: SYS IMP_CDBGL1DCDR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:30]	Way	Cache way	
[29:14]	RES0	Reserved	0b0
[13:6]	Set	Cache set	
[5:3]	Offset	Cache data element offset	
[2:0]	RES0	Reserved	0b0

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C4, #0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C4_0	0b01	0b110	0b1111	0b0100	0b000

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C4, #0, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
```

```
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL1DCDR(X[t]);
```

B.4.10 SYS IMP_CDBGL1ICDR, L1 Instruction Cache Data Read Operation

Read contents of the L1 Instruction Cache Data Memory.

The 40 bits of cache data are written to AArch64-IMP_CDBGDR0_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 B-46: AArch64_sys_imp_cdbgl1icdr bit assignments

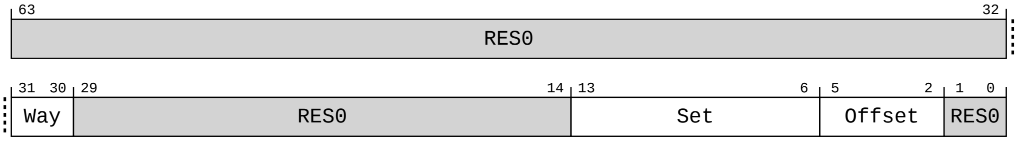


Table B-124: SYS IMP_CDBGL1ICDR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:30]	Way	Cache way	
[29:14]	RES0	Reserved	0b0
[13:6]	Set	Cache set	
[5:2]	Offset	Cache data element offset	
[1:0]	RES0	Reserved	0b0

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C4, #1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C4_1	0b01	0b110	0b1111	0b0100	0b001

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C4, #1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL1ICDR(X[t]);
```

B.4.11 SYS IMP_CDBGL2TR2, L2 TLB Read Operation 2

Read contents of the Level 2 TLB Memory.

Bits [191:128] of the TLB data are written to AArch64-IMP_CDBGDR0_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 B-47: AArch64_sys_imp_cdbgl2tr2 bit assignments

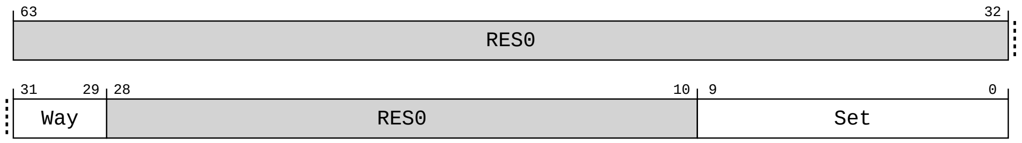


Table B-126: SYS IMP_CDBGL2TR2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:29]	Way	TLB way	
[28:10]	RES0	Reserved	0b0
[9:0]	Set	TLB set. For a single-CPU configuration sets 0x000-0x07F access the main TLB and sets 0x080-0x0A4 access the TLB for IPA and walk entries. For a dual-core configuration sets 0x000-0x0FF access the main TLB and sets 0x100-0x147 access the TLB for IPA and walk entries.	

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C4, #2, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C4_2	0b01	0b110	0b1111	0b0100	0b010

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C4, #2, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL2TR2(X[t]);
```

B.4.12 SYS IMP_CDBGL2CDR, L2 Cache Data Read Operation

Read contents of the L2 Cache Data Memory.

The 64 bits of cache data are written to AArch64-IMP_CDBGDR0_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 B-48: AArch64_sys_imp_cdbgl2cdr bit assignments

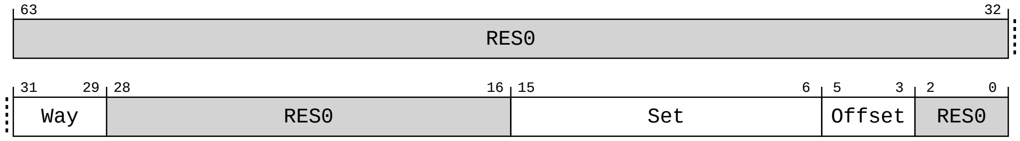


Table B-128: SYS IMP_CDBGL2CDR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:29]	Way	Cache way	
[28:16]	RES0	Reserved	0b0
[15:6]	Set	Cache set	
[5:3]	Offset	Cache data element offset	
[2:0]	RES0	Reserved	0b0

Access

Accesses to this instruction use the following encodings:

SYS #6, C15, C4, #3, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
S1_6_C15_C4_3	0b01	0b110	0b1111	0b0100	0b011

Accessibility

Accesses to this instruction use the following encodings:

SYS #6, C15, C4, #3, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
```

```
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    SYS_IMP_CDBGL2CDR(X[t]);
```

B.5 Identification register summary

The summary table provides an overview of identification registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table B-130: 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_PFR0_EL1	3	C0	0	C1	0	See individual bit resets.	64-bit	AArch32 Processor Feature Register 0
ID_PFR1_EL1	3	C0	0	C1	1	See individual bit resets.	64-bit	AArch32 Processor Feature Register 1
ID_DFR0_EL1	3	C0	0	C1	2	See individual bit resets.	64-bit	AArch32 Debug Feature Register 0
ID_AFR0_EL1	3	C0	0	C1	3	See individual bit resets.	64-bit	AArch32 Auxiliary Feature Register 0
ID_MMFR0_EL1	3	C0	0	C1	4	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 0
ID_MMFR1_EL1	3	C0	0	C1	5	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 1
ID_MMFR2_EL1	3	C0	0	C1	6	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 2
ID_MMFR3_EL1	3	C0	0	C1	7	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 3
ID_ISAR0_EL1	3	C0	0	C2	0	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 0
ID_ISAR1_EL1	3	C0	0	C2	1	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 1
ID_ISAR2_EL1	3	C0	0	C2	2	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 2
ID_ISAR3_EL1	3	C0	0	C2	3	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 3
ID_ISAR4_EL1	3	C0	0	C2	4	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 4
ID_ISAR5_EL1	3	C0	0	C2	5	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 5
ID_MMFR4_EL1	3	C0	0	C2	6	See individual bit resets.	64-bit	AArch32 Memory Model Feature Register 4
ID_ISAR6_EL1	3	C0	0	C2	7	See individual bit resets.	64-bit	AArch32 Instruction Set Attribute Register 6
MVFR0_EL1	3	C0	0	C3	0	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 0
MVFR1_EL1	3	C0	0	C3	1	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 1
MVFR2_EL1	3	C0	0	C3	2	See individual bit resets.	64-bit	AArch32 Media and VFP Feature Register 2
ID_PFR2_EL1	3	C0	0	C3	4	See individual bit resets.	64-bit	AArch32 Processor Feature Register 2
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

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
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
CCSIDR_EL1	3	C0	1	C0	0	See individual bit resets.	64-bit	Current Cache Size ID Register
CLIDR_EL1	3	C0	1	C0	1	See individual bit resets.	64-bit	Cache Level ID Register
GMID_EL1	3	C0	1	C0	4	See individual bit resets.	64-bit	Multiple tag transfer ID register
CSSELR_EL1	3	C0	2	C0	0	See individual bit resets.	64-bit	Cache Size Selection 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
IMP_CPUMPMMCR_EL3	3	C15	6	C2	1	`0x0`	64-bit	Global MPMM Configuration Register

B.5.1 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 B-49: AArch64_midr_el1 bit assignments

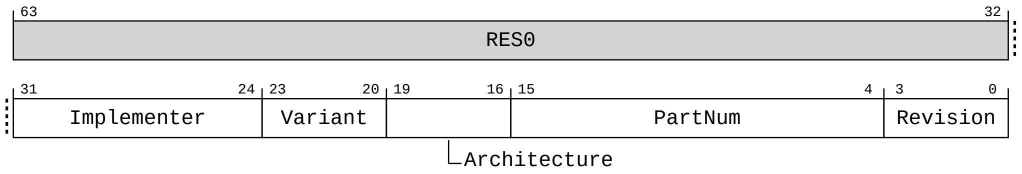


Table B-131: MIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0b0`
[31:24]	Implementer	Indicates the implementer code. This value is: **`01000001`** Arm Limited	
[23:20]	Variant	Indicates the major revision of the product. **`0000`** r0p3	
[19:16]	Architecture	The permitted values of this field are: **`1111`** Architecture is defined by ID registers	
[15:4]	PartNum	An **IMPLEMENTATION DEFINED** primary part number for the device. On processors implemented by Arm, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently. **`110101000110`** Cortex-A510	
[3:0]	Revision	Indicates the minor revision of the product. **`0011`** r0p3	

Access

MRS <Xt>, 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);
```

```
elsif PSTATE.EL == EL1 then
    if EL2Enabled() then
        return VPIDR_EL2;
    else
        return MIDR_EL1;
elsif PSTATE.EL == EL2 then
    return MIDR_EL1;
elsif PSTATE.EL == EL3 then
    return MIDR_EL1;
```

B.5.2 MPIDR_EL1, Multiprocessor Affinity Register

In a multiprocessor system, provides an additional PE identification mechanism for scheduling purposes.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-50: AArch64_mpidr_el1 bit assignments

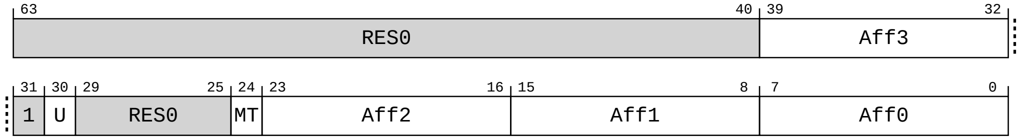


Table B-133: MPIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	`0b0`
[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: **0** Processor is part of a multiprocessor system.	

Bits	Name	Description	Reset
[29:25]	RES0	Reserved	`0b0`
[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: **1** 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. Identification number for each core 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. **00000000** Thread 0 Cortex-A510 is single-threaded.	

Access

MRS <Xt>, MPIDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MPIDR_EL1	`0b11`	`0b000`	`0b0000`	`0b0000`	`0b101`

Accessibility

MRS <Xt>, MPIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() then
        return VMPIDR_EL2;
    else
        return MPIDR_EL1;
elsif PSTATE.EL == EL2 then
    return MPIDR_EL1;
elsif PSTATE.EL == EL3 then
    return MPIDR_EL1;
```

B.5.3 REVIDR_EL1, Revision ID Register

Provides implementation-specific minor revision information.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-51: AArch64_revidr_el1 bit assignments

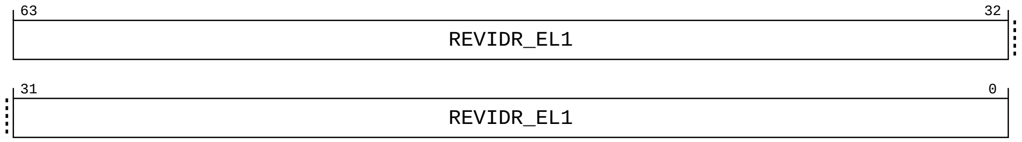


Table B-135: REVIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	REVIDR_EL1	Identifies errata fixes present in this implementation. Refer to the Software Developer's Errata Notice or Product Errata Notice for information on how to interpret this field.	

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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID1 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
```

```
        return REVIDR_EL1;
elsif PSTATE.EL == EL2 then
    return REVIDR_EL1;
elsif PSTATE.EL == EL3 then
    return REVIDR_EL1;
```

B.5.4 ID_PFR0_EL1, AArch32 Processor Feature Register 0

Gives top-level information about the instruction sets supported by the PE in AArch32 state.

Must be interpreted with AArch64-ID_PFR1_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-52: AArch64_id_pfr0_el1 bit assignments

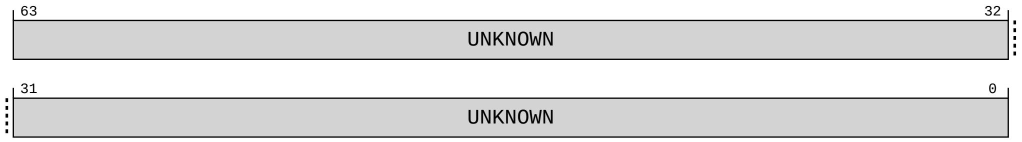


Table B-137: ID_PFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_PFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_PFR0_EL1	0b11	0b000	0b0000	0b0001	0b000

Accessibility

MRS <Xt>, ID_PFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_PFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_PFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_PFR0_EL1;
```

B.5.5 ID_PFR1_EL1, AArch32 Processor Feature Register 1

Gives information about the AArch32 programmers' model.

Must be interpreted with AArch64-ID_PFR0_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-53: AArch64_id_pfr1_el1 bit assignments

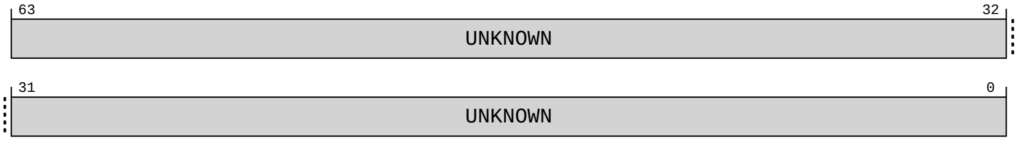


Table B-139: ID_PFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_PFR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_PFR1_EL1	`0b11`	`0b000`	`0b0000`	`0b0001`	`0b001`

Accessibility

MRS <Xt>, ID_PFR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_PFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_PFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_PFR1_EL1;
```

B.5.6 ID_DFR0_EL1, AArch32 Debug Feature Register 0

Provides top level information about the debug system in AArch32 state.

Must be interpreted with the Main ID Register, AArch64-MIDR_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-54: AArch64_id_dfr0_el1 bit assignments

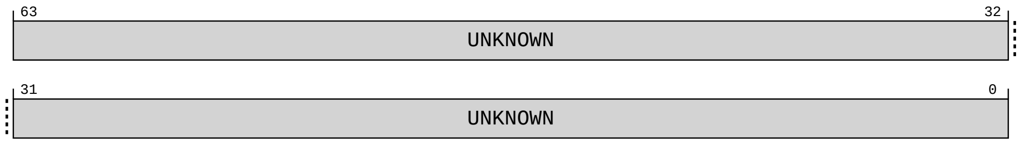


Table B-141: ID_DFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_DFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_DFR0_EL1	0b11	0b000	0b0000	0b0001	0b010

Accessibility

MRS <Xt>, ID_DFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_DFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_DFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_DFR0_EL1;
```

B.5.7 ID_AFR0_EL1, AArch32 Auxiliary Feature Register 0

Provides information about the **IMPLEMENTATION DEFINED** features of the PE in AArch32 state.

Must be interpreted with the Main ID Register, AArch64-MIDR_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-55: AArch64_id_afr0_el1 bit assignments

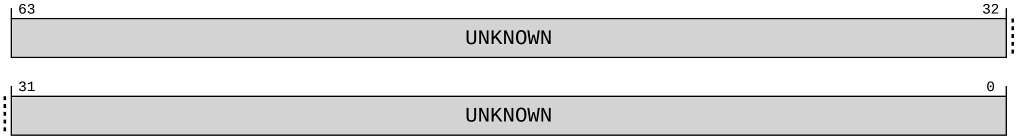


Table B-143: ID_AFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_AFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AFR0_EL1	0b11	0b000	0b0000	0b0001	0b011

Accessibility

MRS <Xt>, ID_AFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AFR0_EL1;
```

B.5.8 ID_MMFR0_EL1, AArch32 Memory Model Feature Register 0

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

Must be interpreted with AArch64-ID_MMFR1_EL1, AArch64-ID_MMFR2_EL1, AArch64-ID_MMFR3_EL1, and AArch64-ID_MMFR4_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-56: AArch64_id_mmfr0_el1 bit assignments

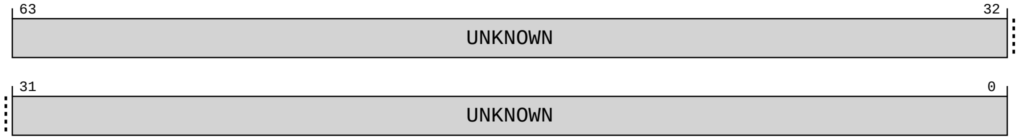


Table B-145: ID_MMFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_MMFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_MMFR0_EL1	0b11	0b000	0b0000	0b0001	0b100

Accessibility

MRS <Xt>, ID_MMFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_MMFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_MMFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_MMFR0_EL1;
```

B.5.9 ID_MMFR1_EL1, AArch32 Memory Model Feature Register 1

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

Must be interpreted with AArch64-ID_MMFR0_EL1, AArch64-ID_MMFR2_EL1, AArch64-ID_MMFR3_EL1, and AArch64-ID_MMFR4_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-57: AArch64_id_mmfr1_el1 bit assignments

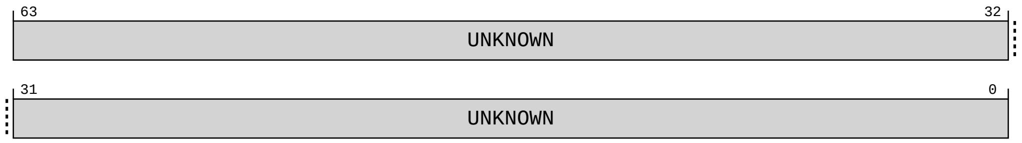


Table B-147: ID_MMFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_MMFR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_MMFR1_EL1	0b11	0b000	0b0000	0b0001	0b101

Accessibility

MRS <Xt>, ID_MMFR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_MMFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_MMFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_MMFR1_EL1;
```

B.5.10 ID_MMFR2_EL1, AArch32 Memory Model Feature Register 2

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

Must be interpreted with AArch64-ID_MMFR0_EL1, AArch64-ID_MMFR1_EL1, AArch64-ID_MMFR3_EL1, and AArch64-ID_MMFR4_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-58: AArch64_id_mmfr2_el1 bit assignments

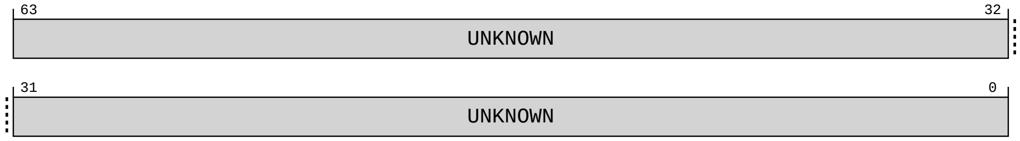


Table B-149: ID_MMFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_MMFR2_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_MMFR2_EL1	0b11	0b000	0b0000	0b0001	0b110

Accessibility

MRS <Xt>, ID_MMFR2_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_MMFR2_EL1;
elsif PSTATE.EL == EL2 then
    return ID_MMFR2_EL1;
elsif PSTATE.EL == EL3 then
    return ID_MMFR2_EL1;
```

B.5.11 ID_MMFR3_EL1, AArch32 Memory Model Feature Register 3

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

Must be interpreted with AArch64-ID_MMFR0_EL1, AArch64-ID_MMFR1_EL1, AArch64-ID_MMFR2_EL1, and AArch64-ID_MMFR4_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-59: AArch64_id_mmfr3_el1 bit assignments

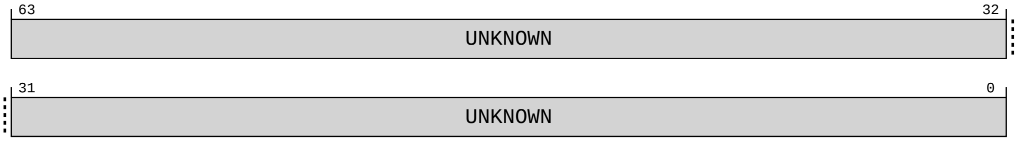


Table B-151: ID_MMFR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_MMFR3_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_MMFR3_EL1	0b11	0b000	0b0000	0b0001	0b111

Accessibility

MRS <Xt>, ID_MMFR3_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_MMFR3_EL1;
elsif PSTATE.EL == EL2 then
    return ID_MMFR3_EL1;
elsif PSTATE.EL == EL3 then
    return ID_MMFR3_EL1;
```

B.5.12 ID_ISAR0_EL1, AArch32 Instruction Set Attribute Register 0

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1, and AArch64-ID_ISAR5_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-60: AArch64_id_isar0_el1 bit assignments

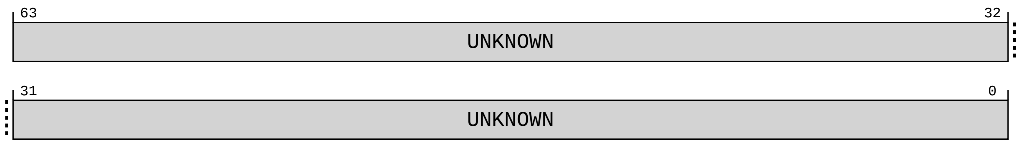


Table B-153: ID_ISAR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_ISAR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR0_EL1	0b11	0b000	0b0000	0b0010	0b000

Accessibility

MRS <Xt>, ID_ISAR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_ISAR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR0_EL1;
```

B.5.13 ID_ISAR1_EL1, AArch32 Instruction Set Attribute Register 1

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1, and AArch64-ID_ISAR5_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-61: AArch64_id_isar1_el1 bit assignments

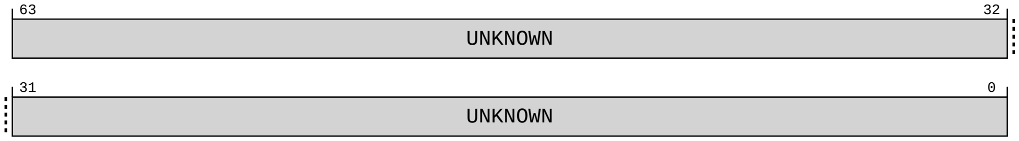


Table B-155: ID_ISAR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_ISAR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR1_EL1	0b11	0b000	0b0000	0b0010	0b001

Accessibility

MRS <Xt>, ID_ISAR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_ISAR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR1_EL1;
```

B.5.14 ID_ISAR2_EL1, AArch32 Instruction Set Attribute Register 2

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1, and AArch64-ID_ISAR5_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-62: AArch64_id_isar2_el1 bit assignments

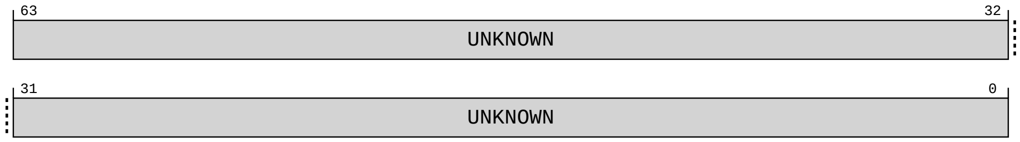


Table B-157: ID_ISAR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_ISAR2_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR2_EL1	0b11	0b000	0b0000	0b0010	0b010

Accessibility

MRS <Xt>, ID_ISAR2_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_ISAR2_EL1;
```

```
elsif PSTATE.EL == EL2 then
    return ID_ISAR2_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR2_EL1;
```

B.5.15 ID_ISAR3_EL1, AArch32 Instruction Set Attribute Register 3

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR4_EL1, and AArch64-ID_ISAR5_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-63: AArch64_id_isar3_el1 bit assignments

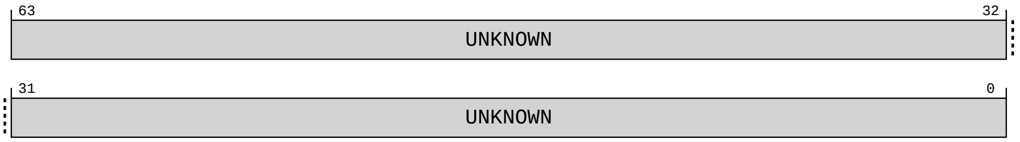


Table B-159: ID_ISAR3_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_ISAR3_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR3_EL1	0b11	0b000	0b0000	0b0010	0b011

Accessibility

MRS <Xt>, ID_ISAR3_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_ISAR3_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR3_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR3_EL1;
```

B.5.16 ID_ISAR4_EL1, AArch32 Instruction Set Attribute Register 4

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, and AArch64-ID_ISAR5_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-64: AArch64_id_isar4_el1 bit assignments

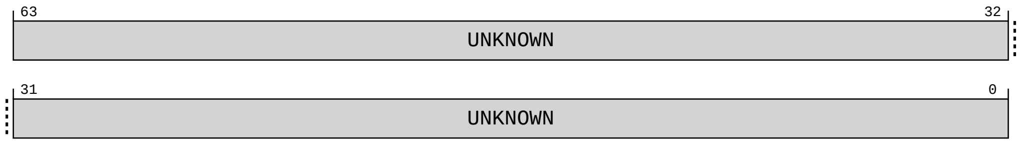


Table B-161: ID_ISAR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_ISAR4_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR4_EL1	0b11	0b000	0b0000	0b0010	0b100

Accessibility

MRS <Xt>, ID_ISAR4_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_ISAR4_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR4_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR4_EL1;
```

B.5.17 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, and AArch64-ID_ISAR4_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-65: AArch64_id_isar5_el1 bit assignments

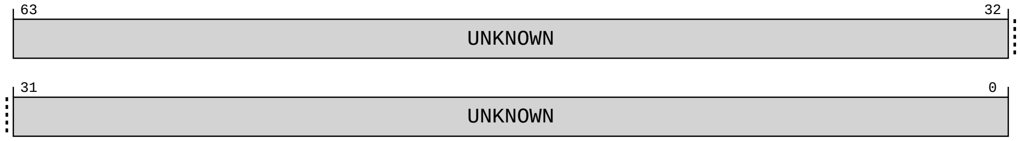


Table B-163: ID_ISAR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_ISAR5_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR5_EL1	0b11	0b000	0b0000	0b0010	0b101

Accessibility

MRS <Xt>, ID_ISAR5_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_ISAR5_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR5_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR5_EL1;
```

B.5.18 ID_MMFR4_EL1, AArch32 Memory Model Feature Register 4

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

Must be interpreted with AArch64-ID_MMFR0_EL1, AArch64-ID_MMFR1_EL1, AArch64-ID_MMFR2_EL1, and AArch64-ID_MMFR3_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-66: AArch64_id_mmfr4_el1 bit assignments

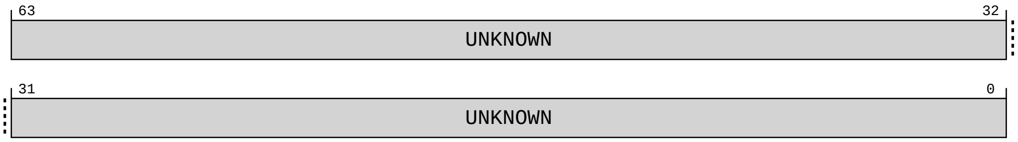


Table B-165: ID_MMFR4_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_MMFR4_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_MMFR4_EL1	0b11	0b000	0b0000	0b0010	0b110

Accessibility

MRS <Xt>, ID_MMFR4_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_MMFR4_EL1;
elsif PSTATE.EL == EL2 then
    return ID_MMFR4_EL1;
elsif PSTATE.EL == EL3 then
    return ID_MMFR4_EL1;
```

B.5.19 ID_ISAR6_EL1, AArch32 Instruction Set Attribute Register 6

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, AArch64-ID_ISAR4_EL1 and AArch64-ID_ISAR5_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-67: AArch64_id_isar6_el1 bit assignments

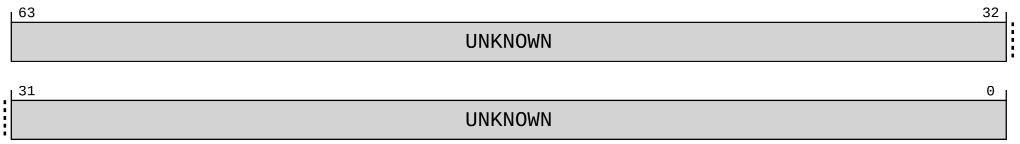


Table B-167: ID_ISAR6_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_ISAR6_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR6_EL1	`0b11`	`0b000`	`0b0000`	`0b0010`	`0b111`

Accessibility

MRS <Xt>, ID_ISAR6_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_ISAR6_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR6_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR6_EL1;
```

B.5.20 MVFR0_EL1, AArch32 Media and VFP Feature Register 0

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with AArch64-MVFR1_EL1 and AArch64-MVFR2_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-68: AArch64_mvfr0_el1 bit assignments

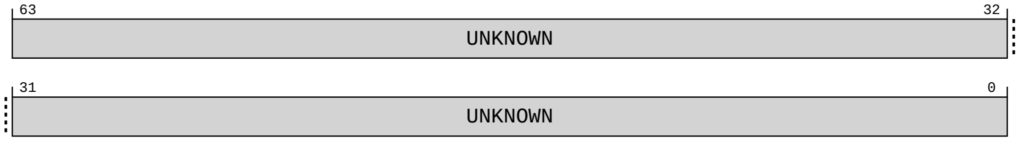


Table B-169: MVFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, MVFR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MVFR0_EL1	0b11	0b000	0b0000	0b0011	0b000

Accessibility

MRS <Xt>, MVFR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return MVFR0_EL1;
elsif PSTATE.EL == EL2 then
    return MVFR0_EL1;
elsif PSTATE.EL == EL3 then
    return MVFR0_EL1;
```

B.5.21 MVFR1_EL1, AArch32 Media and VFP Feature Register 1

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with AArch64-MVFR0_EL1 and AArch64-MVFR2_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-69: AArch64_mvfr1_el1 bit assignments

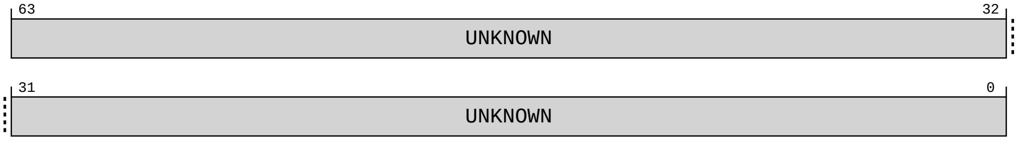


Table B-171: MVFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, MVFR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MVFR1_EL1	0b11	0b000	0b0000	0b0011	0b001

Accessibility

MRS <Xt>, MVFR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return MVFR1_EL1;
```

```
elsif PSTATE.EL == EL2 then
    return MVFR1_EL1;
elsif PSTATE.EL == EL3 then
    return MVFR1_EL1;
```

B.5.22 MVFR2_EL1, AArch32 Media and VFP Feature Register 2

Describes the features provided by the AArch32 Advanced SIMD and Floating-point implementation.

Must be interpreted with AArch64-MVFR0_EL1 and AArch64-MVFR1_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-70: AArch64_mvfr2_el1 bit assignments

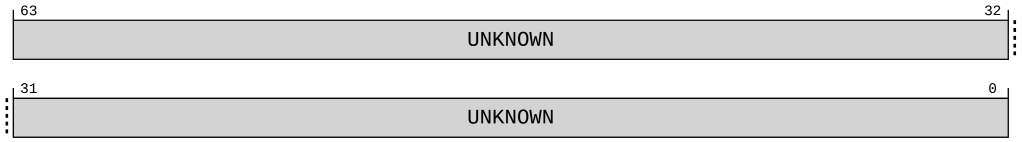


Table B-173: MVFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, MVFR2_EL1

<systemreg>	op0	op1	CRn	CRm	op2
MVFR2_EL1	0b11	0b000	0b0000	0b0011	0b010

Accessibility

MRS <Xt>, MVFR2_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return MVFR2_EL1;
elsif PSTATE.EL == EL2 then
    return MVFR2_EL1;
elsif PSTATE.EL == EL3 then
    return MVFR2_EL1;
```

B.5.23 ID_PFR2_EL1, AArch32 Processor Feature Register 2

Gives information about the AArch32 programmers' model.

Must be interpreted with AArch64-ID_PFR0_EL1 and AArch64-ID_PFR1_EL1.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-71: AArch64_id_pfr2_el1 bit assignments

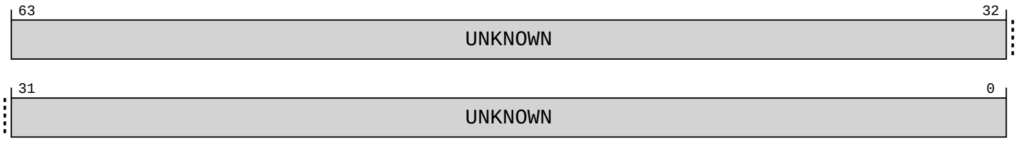


Table B-175: ID_PFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	UNKNOWN	Reserved	

Access

MRS <Xt>, ID_PFR2_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_PFR2_EL1	0b11	0b000	0b0000	0b0011	0b100

Accessibility

MRS <Xt>, ID_PFR2_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_PFR2_EL1;
elsif PSTATE.EL == EL2 then
    return ID_PFR2_EL1;
elsif PSTATE.EL == EL3 then
    return ID_PFR2_EL1;
```

B.5.24 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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-72: AArch64_id_aa64pfr0_el1 bit assignments

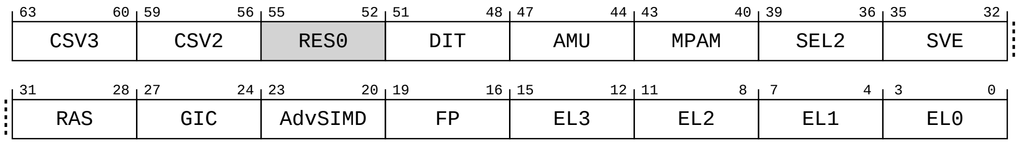


Table B-177: ID_AA64PFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	CSV3	Speculative use of faulting data. Defined values are: `0001` 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: `0010` 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	`0b0`
[51:48]	DIT	Data Independent Timing. Defined values are: `0001` 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: `0001` AMUv1 for Armv8.4 is implemented.	
[43:40]	MPAM	Indicates support for MPAM Extension. Defined values are: `0001` MPAM is implemented.	
[39:36]	SEL2	Secure EL2. Defined values are: `0001` Secure EL2 is implemented.	

Bits	Name	Description	Reset
[35:32]	SVE	Scalable Vector Extension. Defined values are: **0001** SVE architectural state and programmers' model are implemented.	
[31:28]	RAS	RAS Extension version. Defined values are: **0010** ARMv8.4-RAS present.	
[27:24]	GIC	System register GIC CPU interface. Defined values are: **0000** GIC CPU interface system registers not implemented. This value is reported when the GICCDISABLE input is HIGH. **0011** System register interface to version 4.1 of the GIC CPU interface is supported. This value is reported when the GICCDISABLE input is LOW.	
[23:20]	AdvSIMD	Advanced SIMD. Defined values are: **0001** Advanced SIMD is implemented, including support for the following SISD and SIMD operations: * Integer byte, halfword, word and doubleword element operations. * Half-precision, single-precision and double-precision floating-point arithmetic. * Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.	
[19:16]	FP	Floating-point. Defined values are: **0001** Floating-point is implemented, and includes support for: * Half-precision, single-precision and double-precision floating-point types. * Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.	
[15:12]	EL3	EL3 Exception level handling. Defined values are: **0001** EL3 can be executed in AArch64 state only.	
[11:8]	EL2	EL2 Exception level handling. Defined values are: **0001** EL2 can be executed in AArch64 state only.	
[7:4]	EL1	EL1 Exception level handling. Defined values are: **0001** EL1 can be executed in AArch64 state only.	
[3:0]	EL0	EL0 Exception level handling. Defined values are: **0001** EL0 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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64PFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64PFR0_EL1;
```

B.5.25 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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-73: AArch64_id_aa64pfr1_el1 bit assignments

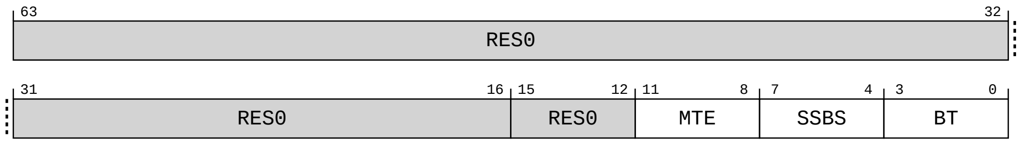


Table B-179: ID_AA64PFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:12]	RES0	Reserved	`0b0`

[11:8]	MTE	Support for the Memory Tagging Extension. Defined values are:
		`0001`
		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.
		`0010`
		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: `0010` 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	
[3:0]	BT	Branch Target Identification mechanism support in AArch64 state. Defined values are: `0001` The Branch Target Identification mechanism is implemented.	

Access

MRS <Xt>, ID_AA64PFR1_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64PFR1_EL1	`0b11`	`0b000`	`0b0000`	`0b0100`	`0b001`

Accessibility

MRS <Xt>, ID_AA64PFR1_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64PFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64PFR1_EL1;
```

```
elsif PSTATE.EL == EL3 then
    return ID_AA64PFR1_EL1;
```

B.5.26 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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-74: AArch64_id_aa64zfr0_el1 bit assignments

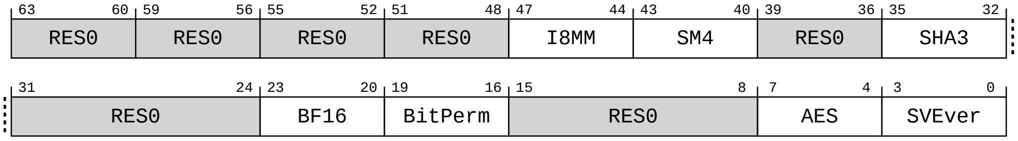


Table B-181: ID_AA64ZFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	`0b0`
[47:44]	I8MM	Indicates support for SVE Int8 matrix multiplication instructions. Defined values are: **`0001`** SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.	

Bits	Name	Description	Reset
[43:40]	SM4	Indicates support for SVE2 SM4 instructions. Defined values are: **0000** SVE2 SM4 instructions are not implemented. This value is reported when Cryptographic Extensions are not implemented or are disabled. **0001** SVE2 SM4E and SM4EKEY instructions are implemented. This value is reported when Cryptographic Extensions are implemented and enabled.	
[39:36]	RES0	Reserved	0b0
[35:32]	SHA3	Indicates support for the SVE2 SHA-3 instruction. Defined values are: **0000** SVE2 SHA-3 instructions are not implemented. This value is reported when Cryptographic Extensions are not implemented or are disabled. **0001** SVE2 RAX1 instruction is implemented. This value is reported when Cryptographic Extensions are implemented and enabled.	
[31:24]	RES0	Reserved	0b0
[23:20]	BF16	Indicates support for SVE BFloat16 instructions. Defined values are: **0001** BFCVT, BFCVTNT, BFDOT, BFMLALB, BFMLALT, and BFMMLA instructions are implemented.	
[19:16]	BitPerm	Indicates support for SVE2 bit permute instructions. Defined values are: **0001** SVE2 BDEP, BEXT and BGRP instructions are implemented.	
[15:8]	RES0	Reserved	0b0
[7:4]	AES	Indicates support for SVE2-AES instructions. Defined values are: **0000** SVE2-AES instructions are not implemented. This value is reported when Cryptographic Extensions are not implemented or are disabled. **0010** 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: **0001** 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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64ZFR0_EL1;
```

B.5.27 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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-75: AArch64_id_aa64dfr0_el1 bit assignments

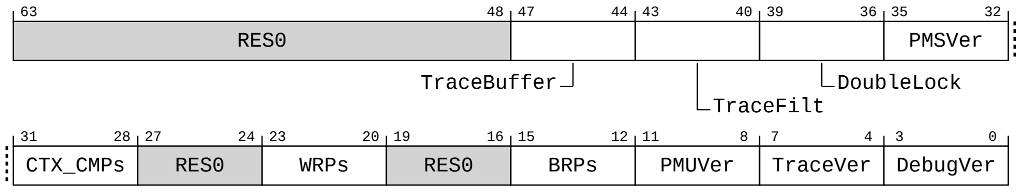


Table B-183: ID_AA64DFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	`0b0`
[47:44]	TraceBuffer	Trace Buffer Extension version. Defined values are: **`0001`** Trace Buffer Extension implemented.	
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. Defined values are: **`0001`** Armv8.4 Self-hosted Trace Extension implemented.	
[39:36]	DoubleLock	OS Double Lock implemented. Defined values are: **`1111`** OS Double Lock not implemented. AArch64-OSDLR_EL1 is RAZ/WI.	
[35:32]	PMSVer	Statistical Profiling Extension version. Defined values are: **`0000`** Statistical Profiling Extension not implemented.	
[31:28]	CTX_CMPs	Number of breakpoints that are context-aware, minus 1. These are the highest numbered breakpoints. **`0001`** Two context-aware breakpoints are included	
[27:24]	RES0	Reserved	`0b0`
[23:20]	WRPs	Number of watchpoints, minus 1. The value of 0b0000 is reserved. **`0011`** Four watchpoints	
[19:16]	RES0	Reserved	`0b0`
[15:12]	BRPs	Number of breakpoints, minus 1. The value of 0b0000 is reserved. **`0101`** Six breakpoints	
[11:8]	PMUVer	Performance Monitors Extension version. Defined value is: **`0110`** 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: **`0001`** PE trace unit System registers implemented.	
[3:0]	DebugVer	Debug architecture version. Indicates presence of Armv8 debug architecture. Defined values are: **`1001`** 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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64DFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64DFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64DFR0_EL1;
```

B.5.28 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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

0x0

Bit descriptions

Figure B-76: AArch64_id_aa64dfr1_el1 bit assignments

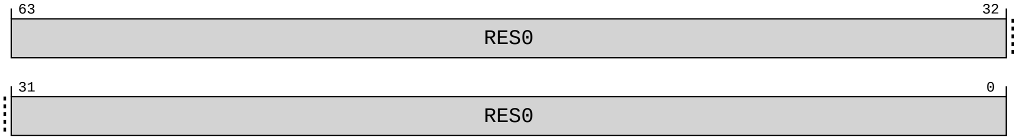


Table B-185: ID_AA64DFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	`0b0`

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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64DFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64DFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64DFR1_EL1;
```

B.5.29 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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

`0x0`

Bit descriptions

Figure B-77: AArch64_id_aa64afr0_el1 bit assignments

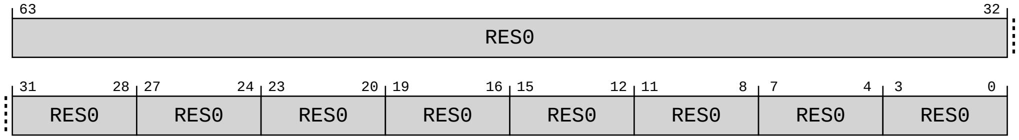


Table B-187: ID_AA64AFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64AFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64AFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64AFR0_EL1;
```

B.5.30 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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

0x0

Bit descriptions

Figure B-78: AArch64_id_aa64afr1_el1 bit assignments

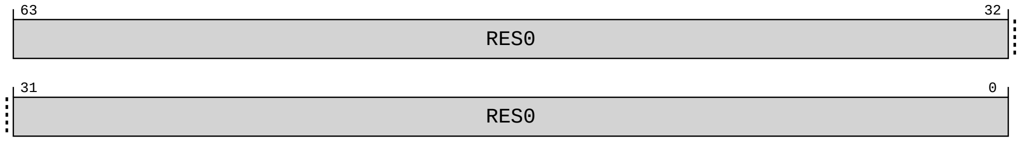


Table B-189: ID_AA64AFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64AFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64AFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64AFR1_EL1;
```

B.5.31 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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-79: AArch64_id_aa64isar0_el1 bit assignments

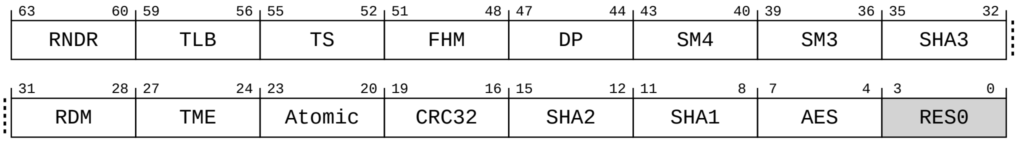


Table B-191: ID_AA64ISAR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	RNDR	Indicates support for Random Number instructions in AArch64 state. Defined values are: `0000` No Random Number instructions are implemented.	
[59:56]	TLB	Indicates support for Outer Shareable and TLB range maintenance instructions. Defined values are: `0010` Outer Shareable and TLB range maintenance instructions are implemented.	
[55:52]	TS	Indicates support for flag manipulation instructions. Defined values are: `0010` CFINV, RMIF, SETF16, SETF8, AXFLAG, and XAFLAG instructions are implemented.	
[51:48]	FHM	Indicates support for FMLAL and FMLSL instructions. Defined values are: `0001` FMLAL and FMLSL instructions are implemented.	

Bits	Name	Description	Reset
[47:44]	DP	Indicates support for Dot Product instructions in AArch64 state. Defined values are: **`0001`** UDOT and SDOT instructions implemented.	
[43:40]	SM4	Indicates support for SM4 instructions in AArch64 state. Defined values are: **`0000`** No SM4 instructions implemented. This value is reported when Cryptographic Extensions are not implemented or are disabled. **`0001`** SM4E and SM4EKEY instructions implemented. This value is reported when Cryptographic Extensions are implemented and enabled.	
[39:36]	SM3	Indicates support for SM3 instructions in AArch64 state. Defined values are: **`0000`** No SM3 instructions implemented. This value is reported when Cryptographic Extensions are not implemented or are disabled. **`0001`** SM3SS1, SM3TT1A, SM3TT1B, SM3TT2A, SM3TT2B, SM3PARTW1, and SM3PARTW2 instructions implemented. This value is reported when Cryptographic Extensions are implemented and enabled.	
[35:32]	SHA3	Indicates support for SHA3 instructions in AArch64 state. Defined values are: **`0000`** No SHA3 instructions implemented. This value is reported when Cryptographic Extensions are not implemented or are disabled. **`0001`** EOR3, RAX1, XAR, and BCAX instructions implemented. This value is reported when Cryptographic Extensions are implemented and enabled.	
[31:28]	RDM	Indicates support for SQRDMLAH and SQRDMLSH instructions in AArch64 state. Defined values are: **`0001`** SQRDMLAH and SQRDMLSH instructions implemented.	
[27:24]	TME	Indicates support for TME instructions. Defined values are: **`0000`** TME instructions are not implemented.	
[23:20]	Atomic	Indicates support for Atomic instructions in AArch64 state. Defined values are: **`0010`** LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions implemented.	
[19:16]	CRC32	CRC32 instructions implemented in AArch64 state. Defined values are: **`0001`** CRC32B, CRC32H, CRC32W, CRC32X, CRC32CB, CRC32CH, CRC32CW, and CRC32CX instructions implemented.	

Bits	Name	Description	Reset

[15:12]	SHA2	SHA2 instructions implemented in AArch64 state. Defined values are:
		`0000`
		No SHA2 instructions implemented. This value is reported when Cryptographic Extensions are not implemented or are disabled.
		`0010`
		SHA256H, SHA256H2, SHA256SU0, SHA256SU1, SHA512H, SHA512H2, SHA512SU0, and SHA512SU1 instructions implemented. This value is reported when Cryptographic Extensions are implemented and enabled.
		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:
		`0000`
		No SHA1 instructions implemented. This value is reported when Cryptographic Extensions are not implemented or are disabled.
		`0001`
		SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 instructions implemented. This value is reported when Cryptographic Extensions are implemented and enabled.
		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:
		`0000`
		No AES instructions implemented. This value is reported when Cryptographic Extensions are not implemented or are disabled.
		`0010`
		AESE, AESD, AESMC, and AESIMC instructions are implemented plus PMULL/PMULL2 instructions operating on 64-bit data quantities. This value is reported when Cryptographic Extensions are implemented and enabled.
		When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	

[3:0]	RES0	Reserved	`0b0`

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);
elsif PSTATE.EL == EL1 then
```

```
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64ISAR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64ISAR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64ISAR0_EL1;
```

B.5.32 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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-80: AArch64_id_aa64isar1_el1 bit assignments

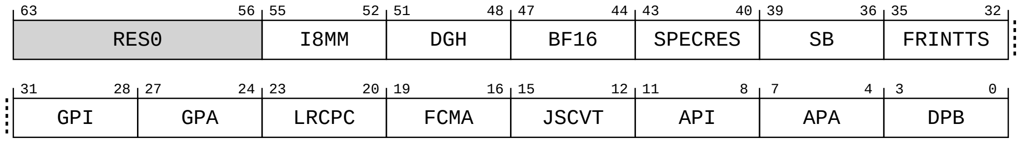


Table B-193: ID_AA64ISAR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	`0b0`
[55:52]	I8MM	Indicates support for Advanced SIMD and Floating-point Int8 matrix multiplication instructions in AArch64 state. Defined values of this field are: **`0001`** SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.	

Bits	Name	Description	Reset
[51:48]	DGH	Indicates support for the Data Gathering Hint instruction. Defined values are: **0000** Data Gathering Hint is not implemented.	
[47:44]	BF16	Indicates support for Advanced SIMD and Floating-point BFloat16 instructions in AArch64 state. Defined values are: **0001** BFDOT, BFMLAL, BFMLAL2, BFMMLA, BFCVT, and BFCVT2 instructions are implemented.	
[43:40]	SPECRES	Indicates support for prediction invalidation instructions in AArch64 state. Defined values are: **0001** CFP RCTX, DVP RCTX, and CPP RCTX instructions are implemented.	
[39:36]	SB	Indicates support for SB instruction in AArch64 state. Defined values are: **0001** SB instruction is implemented.	
[35:32]	FRINTTS	Indicates support for the FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented. Defined values are: **0001** FRINT32Z, FRINT32X, FRINT64Z, and FRINT64X instructions are implemented.	
[31:28]	GPI	Indicates support for an **IMPLEMENTATION DEFINED** algorithm is implemented in the PE for generic code authentication in AArch64 state. Defined values are: **0000** 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: **0001** 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: **0010** The LDAPUR*, STLUR*, and LDAPR* instructions are implemented.	
[19:16]	FCMA	Indicates support for complex number addition and multiplication, where numbers are stored in vectors. Defined values are: **0001** 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: **0001** The FJCVTZS instruction is implemented.	
[11:8]	API	Indicates whether an **IMPLEMENTATION DEFINED** algorithm is implemented in the PE for address authentication, in AArch64 state. This applies to all Pointer Authentication instructions other than the PACGA instruction. Defined values are: **0000** Address Authentication using an **IMPLEMENTATION DEFINED** algorithm is not implemented.	

Bits	Name	Description	Reset
[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: **0101** 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 DC CVADP instructions in AArch64 state. Defined values are: **0010** 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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64ISAR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64ISAR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64ISAR1_EL1;
```

B.5.33 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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-81: AArch64_id_aa64mmfr0_el1 bit assignments

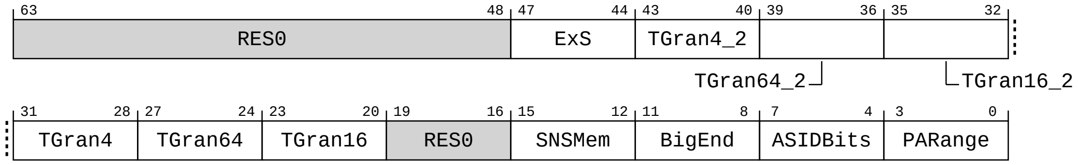


Table B-195: ID_AA64MMFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	0b0
[47:44]	ExS	Indicates support for disabling context synchronizing exception entry and exit. Defined values are: `0000` All exception entries and exits are context synchronization events.	
[43:40]	TGran4_2	Indicates support for 4KB memory granule size for stage 2. Defined values are: `0010` 4KB granule supported at stage 2.	
[39:36]	TGran64_2	Indicates support for 64KB memory granule size for stage 2. Defined values are: `0010` 64KB granule supported at stage 2.	
[35:32]	TGran16_2	Indicates support for 16KB memory granule size for stage 2. Defined values are: `0010` 16KB granule supported at stage 2	
[31:28]	TGran4	Indicates support for 4KB memory translation granule size. Defined values are: `0000` 4KB granule supported.	
[27:24]	TGran64	Indicates support for 64KB memory translation granule size. Defined values are: `0000` 64KB granule supported.	
[23:20]	TGran16	Indicates support for 16KB memory translation granule size. Defined values are: `0001` 16KB granule supported.	
[19:16]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[15:12]	SNSMem	Indicates support for a distinction between Secure and Non-secure Memory. Defined values are: `0001` Does support a distinction between Secure and Non-secure Memory.	
[11:8]	BigEnd	Indicates support for mixed-endian configuration. Defined values are: `0001` Mixed-endian support. The 'SCTLR_ELx'.EE and AArch64-SCTLR_EL1.E0E bits can be configured.	
[7:4]	ASIDBits	Number of ASID bits. Defined values are: `0010` 16 bits.	
[3:0]	PARange	Physical Address range supported. Defined values are: `0010` 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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64MMFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64MMFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64MMFR0_EL1;
```

B.5.34 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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-82: AArch64_id_aa64mmfr1_el1 bit assignments

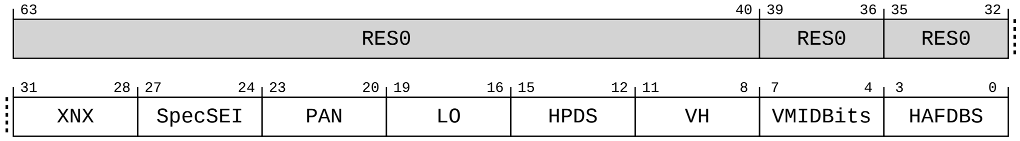


Table B-197: ID_AA64MMFR1_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0b0`
[31:28]	XNX	Indicates support for execute-never control distinction by Exception level at stage 2. Defined values are: **`0001`** 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: **`0001`** The PE might generate 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: **`0010`** PAN supported and `rAT S1E1RP` and `rAT S1E1WP` instructions supported.	
[19:16]	LO	LORegions. Indicates support for LORegions. Defined values are: **`0001`** LORegions supported.	
[15:12]	HPDS	Hierarchical Permission Disables. Indicates support for disabling hierarchical controls in translation tables. Defined values are: **`0010`** 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.	

Bits	Name	Description	Reset
[11:8]	VH	Virtualization Host Extensions. Defined values are: **`0001`** Virtualization Host Extensions supported.	
[7:4]	VMIDBits	Number of VMID bits. Defined values are: **`0010`** 16 bits	
[3:0]	HAFDBS	Hardware updates to Access flag and Dirty state in translation tables. Defined values are: **`0010`** 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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64MMFR1_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64MMFR1_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64MMFR1_EL1;
```

B.5.35 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

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-83: AArch64_id_aa64mmfr2_el1 bit assignments

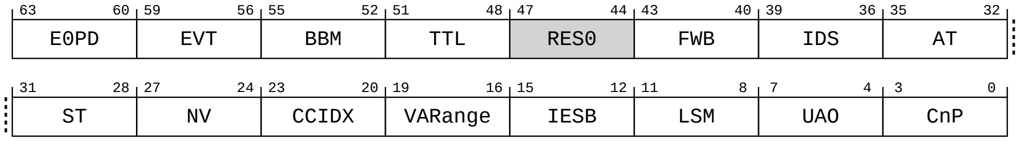


Table B-199: ID_AA64MMFR2_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	E0PD	Indicates support for the E0PD mechanism. Defined values are: **0001** E0PDx 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: **0010** AArch64-HCR_EL2.{TTLBOS, TTLBIS, TOCU, TICAB, TID4} traps are supported.	
[55:52]	BBM	Allows identification of the requirements of the hardware to have break-before-make sequences when changing block size for a translation. **0010** Level 2 support for changing block size is supported.	
[51:48]	TTL	Indicates support for TTL field in address operations. Defined values are: **0001** TLB maintenance instructions by address have bits[47:44] holding the TTL field.	
[47:44]	RES0	Reserved	0b0
[43:40]	FWB	Indicates support for AArch64-HCR_EL2.FWB. Defined values are: **0001** 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: **0001** All exceptions generated by an AArch64 read access to the feature ID space are reported by ESR_ELx.EC == 0x18.	

Bits	Name	Description	Reset
[35:32]	AT	Identifies support for unaligned single-copy atomicity and atomic functions. Defined values are: `0001` 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: `0001` The maximum value of the TCR_ELx.{T0SZ,T1SZ} and VTCR_EL2.T0SZ fields is 48 for 4KB and 16KB granules, and 47 for 64KB granules.	
[27:24]	NV	Nested Virtualization. If EL2 is implemented, indicates support for the use of nested virtualization. Defined values are: `0000` Nested virtualization is not supported.	
[23:20]	CCIDX	Support for the use of revised AArch64-CCSIDR_EL1 register format. Defined values are: `0001` 64-bit format implemented for all levels of the CCSIDR_EL1.	
[19:16]	VARange	Indicates support for a larger virtual address. Defined values are: `0000` VMSAv8-64 supports 48-bit VAs.	
[15:12]	IESB	Indicates support for the IESB bit in the SCTLR_ELx registers. Defined values are: `0001` IESB bit in the SCTLR_ELx registers is supported.	
[11:8]	LSM	Indicates support for LSMAOE and nTLSMD bits in AArch64-SCTLR_EL1 and AArch64-SCTLR_EL2. Defined values are: `0000` LSMAOE and nTLSMD bits not supported.	
[7:4]	UAO	User Access Override. Defined values are: `0001` UAO supported.	
[3:0]	CnP	Indicates support for Common not Private translations. Defined values are: `0001` 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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64MMFR2_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64MMFR2_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64MMFR2_EL1;
```

B.5.36 CCSIDR_EL1, Current Cache Size ID Register

Provides information about the architecture of the currently selected cache.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

The parameters NumSets, Associativity, and LineSize in these registers define the architecturally visible parameters that are required for the cache maintenance by Set/Way instructions. They are not guaranteed to represent the actual microarchitectural features of a design. You cannot make any inference about the actual sizes of caches based on these parameters.

Figure B-84: AArch64_ccsidr_el1 bit assignments

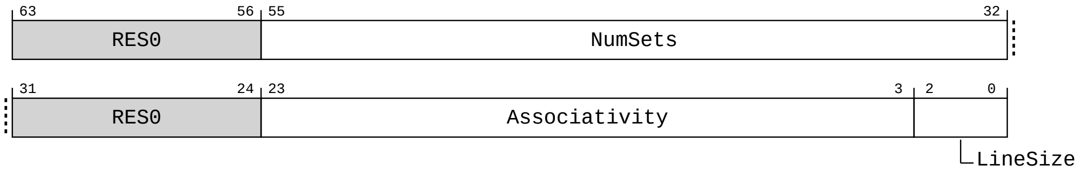


Table B-201: CCSIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:56]	RES0	Reserved	0b0
[55:32]	NumSets	(Number of sets in cache) - 1, therefore a value of 0 indicates 1 set in the cache. The number of sets does not have to be a power of 2.	

Bits	Name	Description	Reset
[31:24]	RES0	Reserved	`0b0`
[23:3]	Associativity	(Associativity of cache) - 1, therefore a value of 0 indicates an associativity of 1. The associativity does not have to be a power of 2.	
[2:0]	LineSize	(Log$_2$(Number of bytes in cache line)) - 4. For example: For a line length of 16 bytes: Log$_2$(16) = 4, LineSize entry = 0. This is the minimum line length. For a line length of 32 bytes: Log$_2$(32) = 5, LineSize entry = 1. When ARMv8.5-MemTag is implemented and enabled, where a cache only holds Allocation tags, this field is RES0.	

Access

MRS <Xt>, CCSIDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
CCSIDR_EL1	`0b11`	`0b001`	`0b0000`	`0b0000`	`0b000`

Accessibility

MRS <Xt>, CCSIDR_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TID4 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CCSIDR_EL1;
elsif PSTATE.EL == EL2 then
    return CCSIDR_EL1;
elsif PSTATE.EL == EL3 then
    return CCSIDR_EL1;
```

B.5.37 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 B-85: AArch64_clidr_el1 bit assignments

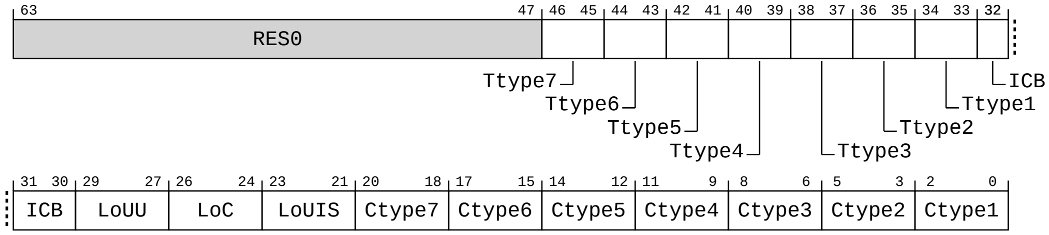


Table B-203: CLIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:47]	RES0	Reserved	0b0
[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. 00 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. 00 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. 00 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. 00 No Tag Cache.	

Bits	Name	Description	Reset

[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.
		`00`
		No Tag Cache. This value is reported if the BROADCASTMTE pin is low or either the Cortex-A510 complex is configured without an L2 cache or the DSU is configured without an L3 cache.
		`10`
		Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines. This value is reported if the BROADCASTMTE pin is high and both the Cortex-A510 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	

[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.
		`00`
		No Tag Cache. This value is reported if the BROADCASTMTE pin is low or both the Cortex-A510 complex is configured without an L2 cache and the DSU is configured without an L3 cache.
		`10`
		Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines. This value is reported if the BROADCASTMTE pin is high and either the Cortex-A510 complex is configured with an L2 cache or the DSU is configured with an L3 cache.	

[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.
		`00`
		No Tag Cache. This value is reported if the BROADCASTMTE pin is low.
		`10`
		Unified Allocation Tag and Data cache, Allocation Tags and Data in unified lines. This value is reported if the BROADCASTMTE pin is high.	

[32:30]	ICB	Inner cache boundary. This field indicates the boundary for caching Inner Cacheable memory regions.
		The possible values are:
		`001`
		L1 cache is the highest Inner Cacheable level. This value is reported if both the Cortex-A510 complex is configured without an L2 cache and the DSU is configured without an L3 cache.
		`010`
		L2 cache is the highest Inner Cacheable level. This value is reported if either but not both of the Cortex-A510 complex is configured with an L2 cache or the DSU is configured with an L3 cache.
		`011`
		L3 cache is the highest Inner Cacheable level. This value is reported if both the Cortex-A510 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	

[29:27]	LoUU	Level of Unification Uniprocessor for the cache hierarchy. `000` Level of Unification Uniprocessor is before the L1 data cache.	

Bits	Name	Description	Reset
[26:24]	LoC	Level of Coherence for the cache hierarchy. **001** Level of Coherency is after the L1 data cache. This value is reported if both the Cortex-A510 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **010** Level of Coherency is after the L2 cache. This value is reported if either but not both of the Cortex-A510 complex is configured with an L2 cache or the DSU is configured with an L3 cache. **011** Level of Coherency is after the L3 cache. This value is reported if both the Cortex-A510 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	
[23:21]	LoUIS	Level of Unification Inner Shareable for the cache hierarchy. **000** Level of Unification Inner Shareable is before the L1 data cache.	
[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: **000** No cache.	
[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: **000** 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: **000** 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: **000** 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: **000** No cache. This value is reported if either the Cortex-A510 complex is configured without an L2 cache or the DSU is configured without an L3 cache. **100** Unified cache. This value is reported if both the Cortex-A510 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	

Bits	Name	Description	Reset

[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:
		`000`
		No cache. This value is reported if both the Cortex-A510 complex is configured without an L2 cache and the DSU is configured without an L3 cache.
		`100`
		Unified cache. This value is reported if either the Cortex-A510 complex is configured with an L2 cache or the DSU is configured with an L3 cache.	

[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: `011` Separate instruction and data caches.	

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);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TID4 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CLIDR_EL1;
elsif PSTATE.EL == EL2 then
    return CLIDR_EL1;
elsif PSTATE.EL == EL3 then
    return CLIDR_EL1;
```

B.5.38 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 B-86: AArch64_gmid_el1 bit assignments

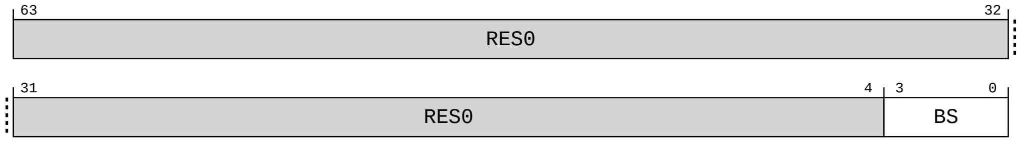


Table B-205: GMID_EL1 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	`0b0`
[3:0]	BS	Log$_2$ of the block size in words. The minimum supported size is 16B (value == 2) and the maximum is 256B (value == 6). **`0100`** 64 bytes.	

Access

MRS <Xt>, GMID_EL1

<systemreg>	op0	op1	CRn	CRm	op2
GMID_EL1	`0b11`	`0b001`	`0b0000`	`0b0000`	`0b100`

Accessibility

MRS <Xt>, GMID_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID5 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return GMID_EL1;
elsif PSTATE.EL == EL2 then
    return GMID_EL1;
elsif PSTATE.EL == EL3 then
    return GMID_EL1;
```

B.5.39 CSSELR_EL1, Cache Size Selection Register

Selects the current Cache Size ID Register, AArch64-CCSIDR_EL1, by specifying the required cache level and the cache type (either instruction or data cache).

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-87: AArch64_csselr_el1 bit assignments

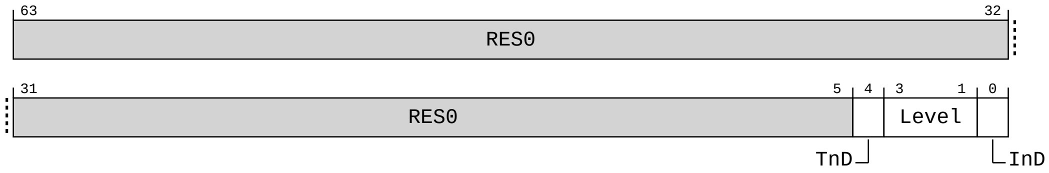


Table B-207: CSSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:5]	RES0	Reserved	0b0
[4]	TnD	Allocation Tag not Data bit. **0** Data, Instruction or Unified cache.	
[3:1]	Level	Cache level of required cache. **000** Level 1 cache. **001** Level 2 cache. **010** Level 3 cache.	

Bits	Name	Description	Reset
[0]	InD	Instruction not Data bit. **0** Data or unified cache. **1** Instruction cache. If CSSELR_EL1.Level is programmed to a cache level that is not implemented, then a read of CSSELR_EL1 is CONSTRAINED UNPREDICTABLE, and returns UNKNOWN values for CSSELR_EL1.{Level, InD}.	

Access

MRS <Xt>, CSSELR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
CSSELR_EL1	0b11	0b010	0b0000	0b0000	0b000

MSR CSSELR_EL1, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
CSSELR_EL1	0b11	0b010	0b0000	0b0000	0b000

Accessibility

MRS <Xt>, CSSELR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TID4 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CSSELR_EL1;
elsif PSTATE.EL == EL2 then
    return CSSELR_EL1;
elsif PSTATE.EL == EL3 then
    return CSSELR_EL1;
```

MSR CSSELR_EL1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.TID4 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        CSSELR_EL1 = X[t];
elsif PSTATE.EL == EL2 then
    CSSELR_EL1 = X[t];
elsif PSTATE.EL == EL3 then
    CSSELR_EL1 = X[t];
```

B.5.40 CTR_EL0, Cache Type Register

Provides information about the architecture of the caches.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure B-88: AArch64_ctr_el0 bit assignments

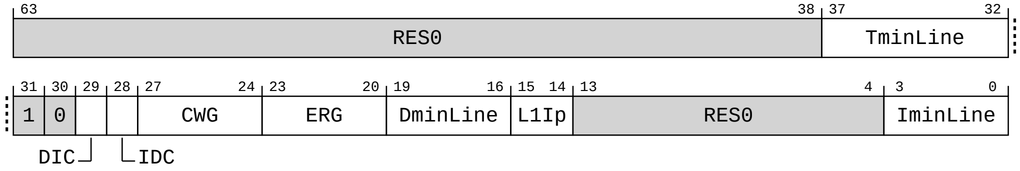


Table B-210: CTR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:38]	RES0	Reserved	`0b0`
[37:32]	TminLine	Tag minimum Line. Log2 of the number of words covered by Allocation Tags in the smallest cache line of all caches which can contain Allocation tags that are controlled by the PE. **`000000`** MTE not supported. This value is reported if the BROADCASTMTE pin is low. **`000100`** 64 bytes. This value is reported if the BROADCASTMTE pin is high.	
[31]	RES1	Reserved	`0b1`
[30]	RES0	Reserved	`0b0`
[29]	DIC	Instruction cache invalidation requirements for data to instruction coherence. **`0`** Instruction cache invalidation to the Point of Unification is required for data to instruction coherence.	
[28]	IDC	Data cache clean requirements for instruction to data coherence. The meaning of this bit is: **`1`** Data cache clean to the Point of Unification is not required for instruction to data coherence.	

Bits	Name	Description	Reset
[27:24]	CWG	Cache writeback granule. Log2 of the number of words of the maximum size of memory that can be overwritten as a result of the eviction of a cache entry that has had a memory location in it modified. **`0100`** 64 bytes.	
[23:20]	ERG	Exclusives reservation granule. Log2 of the number of words of the maximum size of the reservation granule for the Load-Exclusive and Store-Exclusive instructions. **`0100`** 64 bytes.	
[19:16]	DminLine	Log$_2$ of the number of words in the smallest cache line of all the data caches and unified caches that are controlled by the PE. **`0100`** 64 bytes.	
[15:14]	L1Ip	Level 1 instruction cache policy. Indicates the indexing and tagging policy for the L1 instruction cache. Possible values of this field are: **`11`** Physical Index, Physical Tag (PIPT)	
[13:4]	RES0	Reserved	`0b0`
[3:0]	IminLine	Log$_2$ of the number of words in the smallest cache line of all the instruction caches that are controlled by the PE. **`0100`** 64 bytes.	

Access

MRS <Xt>, CTR_EL0

<systemreg>	op0	op1	CRn	CRm	op2
CTR_EL0	`0b11`	`0b011`	`0b0000`	`0b0000`	`0b001`

Accessibility

MRS <Xt>, CTR_EL0

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == 11) && SCTLR_EL1.UCT == 0 then
        if EL2Enabled() && HCR_EL2.TGE == 1 then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != 11 && HCR_EL2.TID2 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == 11 && SCTLR_EL2.UCT == 0 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CTR_EL0;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID2 == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return CTR_EL0;
elsif PSTATE.EL == EL2 then
    return CTR_EL0;
elsif PSTATE.EL == EL3 then
```

```
    return CTR_EL0;
```

B.5.41 DCZID_EL0, Data Cache Zero ID register

Indicates the block size that is written with byte values of 0 by the `DC ZVA` (Data Cache Zero by Address) System instruction.

If ARMv8.5-MemTag is implemented, this register also indicates the granularity at which the `DC GVA` and `DC 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 B-89: AArch64_dczid_el0 bit assignments

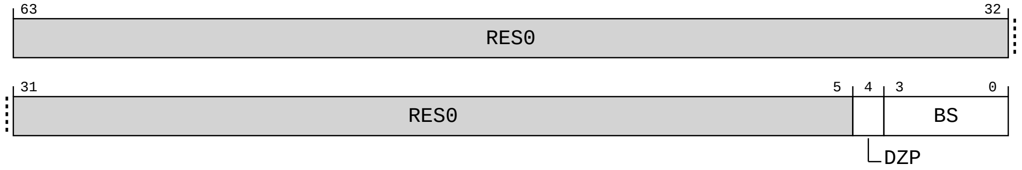


Table B-212: DCZID_EL0 bit descriptions

Bits	Name	Description	Reset
[63:5]	RES0	Reserved	`0b0`
[4]	DZP	Data Zero Prohibited. This field indicates whether use of r`DC ZVA` instructions is permitted or prohibited. If ARMv8.5-MemTag is implemented, this field also indicates whether use of the `DC GVA` and `DC GZVA` instructions are permitted or prohibited. **0** Instructions are permitted. **1** Instructions are prohibited. The value read from this field is governed by the access state and the values of the AArch64-HCR_EL2.TDZ and AArch64-SCTLR_EL1.DZE bits.	

Bits	Name	Description	Reset
[3:0]	BS	Log$_2$ of the block size in words. The maximum size supported is 2KB (value == 9). **0100** 64 bytes.	

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
    return DCZID_EL0;
elsif PSTATE.EL == EL1 then
    return DCZID_EL0;
elsif PSTATE.EL == EL2 then
    return DCZID_EL0;
elsif PSTATE.EL == EL3 then
    return DCZID_EL0;
```

B.5.42 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 B-90: AArch64_mpamidr_el1 bit assignments

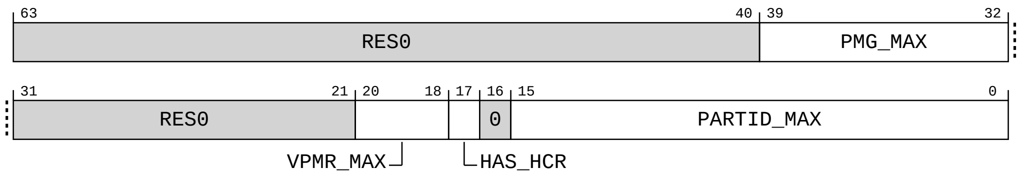


Table B-214: MPAMIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:40]	RES0	Reserved	0b0
[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. **00000001** Max PMG field is 1	
[31:21]	RES0	Reserved	0b0
[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. **001** 2 MPAMVPMn_EL2 registers are implemented	
[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 VPM-R_MAX. Must be 0 if EL2 is not implemented in either security state. **1** MPAM virtualization is supported.	
[16]	RES0	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. **0000000000111111** 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;
```

B.5.43 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 B-91: AArch64_imp_cpucfr_el1 bit assignments

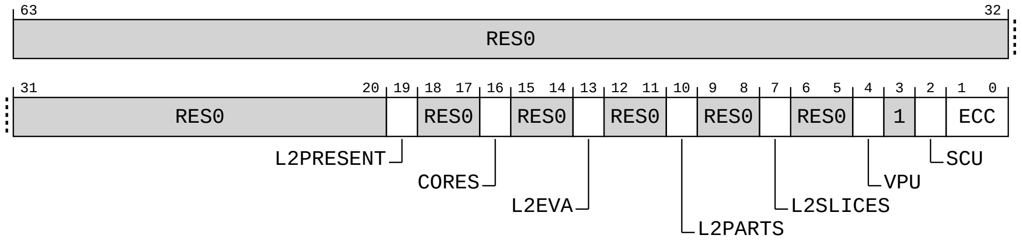


Table B-216: IMP_CPUCFR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	`0b0`
[19]	L2PRESENT	Indicates whether an L2 cache is present in the Cortex-A510 complex containing this core. **0** An L2 cache is not present in the complex. **1** An L2 cache is present in the complex.	
[18:17]	RES0	Reserved	`0b0`

Bits	Name	Description	Reset
[16]	CORES	The number of cores in the Cortex-A510 complex containing this core. **0** One core. **1** Two cores.	
[15:14]	RES0	Reserved	`0b0`
[13]	L2EVA	Indicates whether the L2 cache optimized evict/allocate accesses are implemented. Possible values of this field are: **0** Not implemented. **1** Implemented.	
[12:11]	RES0	Reserved	`0b0`
[10]	L2PARTS	Indicates the configured number of L2 cache partitions. Possible values of this field are: **0** One partition. **1** Two partitions.	
[9:8]	RES0	Reserved	`0b0`
[7]	L2SLICES	Indicates the configured number of L2 cache slices. Possible values of this field are: **0** One slice. **1** Two slices.	
[6:5]	RES0	Reserved	`0b0`
[4]	VPU	Describes the configured VPU datapath width. Possible values of this field are: **0** Two 64-bit datapaths are configured. **1** Two 128-bit datapaths are configured.	
[3]	RES1	Reserved	`0b1`
[2]	SCU	Indicates whether the SCU is present or not. Possible values of this bit are: **0** The SCU is present.	
[1:0]	ECC	Indicates whether ECC is present or not. Possible values of this field are: **00** ECC is not present. **01** ECC is present.	

Access

MRS <Xt>, S3_0_C15_C0_0

<systemreg>	op0	op1	CRn	CRm	op2
S3_0_C15_C0_0	0b11	0b000	0b1111	0b0000	0b000

Accessibility

MRS <Xt>, S3_0_C15_C0_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;
elsif PSTATE.EL == EL2 then
    return IMP_CPUCFR_EL1;
elsif PSTATE.EL == EL3 then
    return IMP_CPUCFR_EL1;
```

B.5.44 IMP_CPUMPMMCR_EL3, Global MPMM Configuration Register

This register is used to change MPMM gears or disable MPMM.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

0x0

Bit descriptions

Figure B-92: AArch64_imp_cpumpmmcr_el3 bit assignments

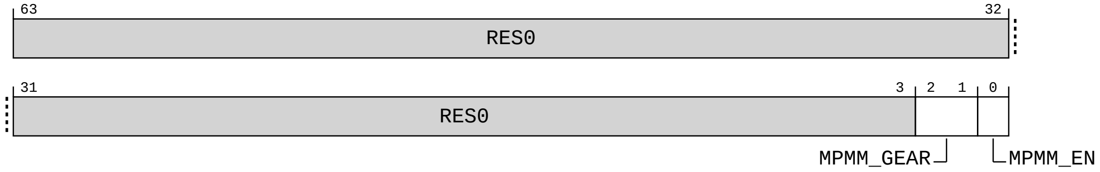


Table B-218: IMP_CPUMPMMCR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[2:1]	MPMM_GEAR	MPMM Gear Select **00** Select MPMM Gear 0. **01** Select MPMM Gear 1. **10** Select MPMM Gear 2.	0b00
[0]	MPMM_EN	MPMM Master Enable **0** MPMM is disabled. **1** MPMM is enabled.	0b0

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;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return IMP_CPUMPMMCR_EL3;
```

MSR S3_6_C15_C2_1, <Xt>

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TIDCP == 1 then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
```

```
elsif PSTATE.EL == EL3 then
    IMP_CPUMPMMCR_EL3 = X[t];
```

B.6 Performance monitors register summary

The summary table provides an overview of performance monitors registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table B-221: 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
PMMIR	-	C9	0	C14	6	-	32-bit	Performance Monitors Machine Identification Register
PMCR_EL0	3	C9	3	C12	0	See individual bit resets	64-bit	Performance Monitors Control Register
PMCR	-	C9	0	C12	0	-	32-bit	Performance Monitors Control Register
PMCEID0_EL0	3	C9	3	C12	6	See individual bit resets	64-bit	Performance Monitors Common Event Identification register 0
PMCEID1_EL0	3	C9	3	C12	7	See individual bit resets	64-bit	Performance Monitors Common Event Identification register 1
PMCEID0	-	C9	0	C12	6	-	32-bit	Performance Monitors Common Event Identification register 0
PMCEID1	-	C9	0	C12	7	-	32-bit	Performance Monitors Common Event Identification register 1

B.6.1 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 B-93: AArch64_pmmir_el1 bit assignments

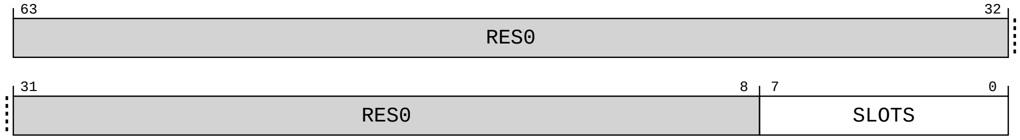


Table B-222: PMMIR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:8]	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 not implemented, this field might read as zero. **00000011** The largest value by which the STALL_SLOT PMU event may increment in one cycle is 3.	

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;
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 PMMIR_EL1;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMMIR_EL1;
elsif PSTATE.EL == EL3 then
    return PMMIR_EL1;
```

B.6.2 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 B-94: AArch64_pmcr_el0 bit assignments

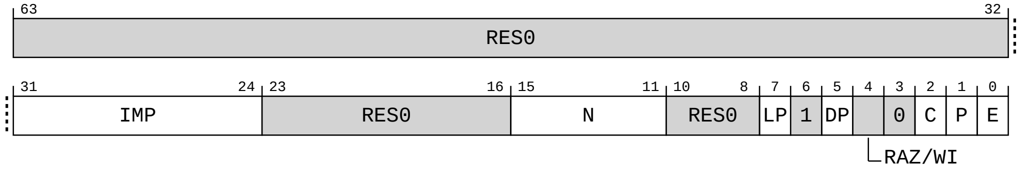


Table B-224: PMCR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:24]	IMP	Implementer code. **00000000** No ID information is present in PMCR/PMCR_EL0. Software must use the MIDR_EL1 to identify the PE.	
[23:16]	RES0	Reserved	0b0
[15:11]	N	Indicates the number of event counters implemented. This value is in the range of 0b00000-0b11111. If the value is 0b00000 then only AArch64-PMCCNTR_EL0 is implemented. If the value is 0b11111 AArch64-PMCCNTR_EL0 and 31 event counters are implemented. When EL2 is implemented and enabled for the current Security state, reads of this field from EL1 and EL0 return the value of AArch64-MDCR_EL2.HPMN. **00110** Six PMU Counters Implemented	
[10:8]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[7]	LP	Long event counter enable. Determines when unsigned overflow is recorded by a counter overflow bit. **0** Event counter overflow on increment that causes unsigned overflow of AArch64-PMEVCNTR<n>_EL0[31:0]. **1** Event counter overflow on increment that causes unsigned overflow of AArch64-PMEVCNTR<n>_EL0[63:0]. If EL2 is implemented and AArch64-MDCR_EL2.HPMN or AArch32-HDCR.HPMN is less than PMCR_EL0.N, this bit does not affect the operation of event counters in the range [AArch32-HDCR.HPMN..(PMCR_EL0.N-1)] or [AArch64-MDCR_EL2.HPMN..(PMCR_EL0.N-1)]. **Note:** 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.	
[6]	RES1	Reserved	`0b1`
[5]	DP	Disable cycle counter when event counting is prohibited. **0** Cycle counting by AArch64-PMCCNTR_EL0 is not affected by this bit. **1** When event counting for counters in the range [0..(AArch64-MDCR_EL2.HPMN-1)] is prohibited, cycle counting by AArch64-PMCCNTR_EL0 is disabled. For more information see 'Prohibiting event counting'.	
[4]	RAZ/WI	Reserved	
[3]	RES0	Reserved	`0b0`
[2]	C	Cycle counter reset. The effects of writing to this bit are: **0** No action. **1** Reset AArch64-PMCCNTR_EL0 to zero. This bit is always RAZ. **Note:** Resetting AArch64-PMCCNTR_EL0 does not change the cycle counter overflow bit. The value of PMCR_EL0.LC is ignored, and bits [63:0] of all affected event counters are reset.	

Bits	Name	Description	Reset
[1]	P	Event counter reset. The effects of writing to this bit are: **0** No action. **1** Reset all event counters accessible in the current Exception level, not including AArch64-PMCCNTR_EL0, to zero. This bit is always RAZ. In EL0 and EL1: If EL2 is implemented and enabled in the current Security state, and AArch64-MDCR_EL2.HPMN is less than PMCR_EL0.N, a write of 1 to this bit does not reset event counters in the range [AArch64-MDCR_EL2.HPMN..(PMCR_EL0.N-1)]. If EL2 is not implemented, EL2 is disabled in the current Security state, or AArch64-MDCR_EL2.HPMN equals PMCR_EL0.N, a write of 1 to this bit resets all the event counters. In EL2 and EL3, a write of 1 to this bit resets all the event counters. **Note:** Resetting the event counters does not change the event counter overflow bits. If ARMv8.5-PMU is implemented, the values of AArch64-MDCR_EL2.HLP and PMCR_EL0.LP are ignored, and bits [63:0] of all affected event counters are reset.	
[0]	E	Enable. **0** All event counters in the range [0..(PMN-1)] and AArch64-PMCCNTR_EL0, are disabled. **1** All event counters in the range [0..(PMN-1)] and AArch64-PMCCNTR_EL0, are enabled by AArch64-PMCNTENSET_EL0. If EL2 is implemented, then: If EL2 is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. If PMN is less than PMCR_EL0.N, this bit does not affect the operation of event counters in the range [PMN..(PMCR_EL0.N-1)]. If EL2 is not implemented, PMN is PMCR_EL0.N. **Note:** 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.	

Access

MRS <Xt>, PMCR_EL0

<systemreg>	op0	op1	CRn	CRm	op2
PMCR_EL0	0b11	0b011	0b1001	0b1100	0b000

MSR PMCR_EL0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
PMCR_EL0	0b11	0b011	0b1001	0b1100	0b000

Accessibility

MRS <Xt>, PMCR_EL0

```
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_EL0;
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_EL0;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCR_EL0;
elsif PSTATE.EL == EL3 then
    return PMCR_EL0;
```

MSR PMCR_EL0, <Xt>

```
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
        PMCR_EL0 = X[t];
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
        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];
```

B.6.3 PMCEID0_EL0, 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.

Note

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

For more information about the common events and the use of the PMCEID<n>_EL0 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 B-95: AArch64_pmceid0_el0 bit assignments

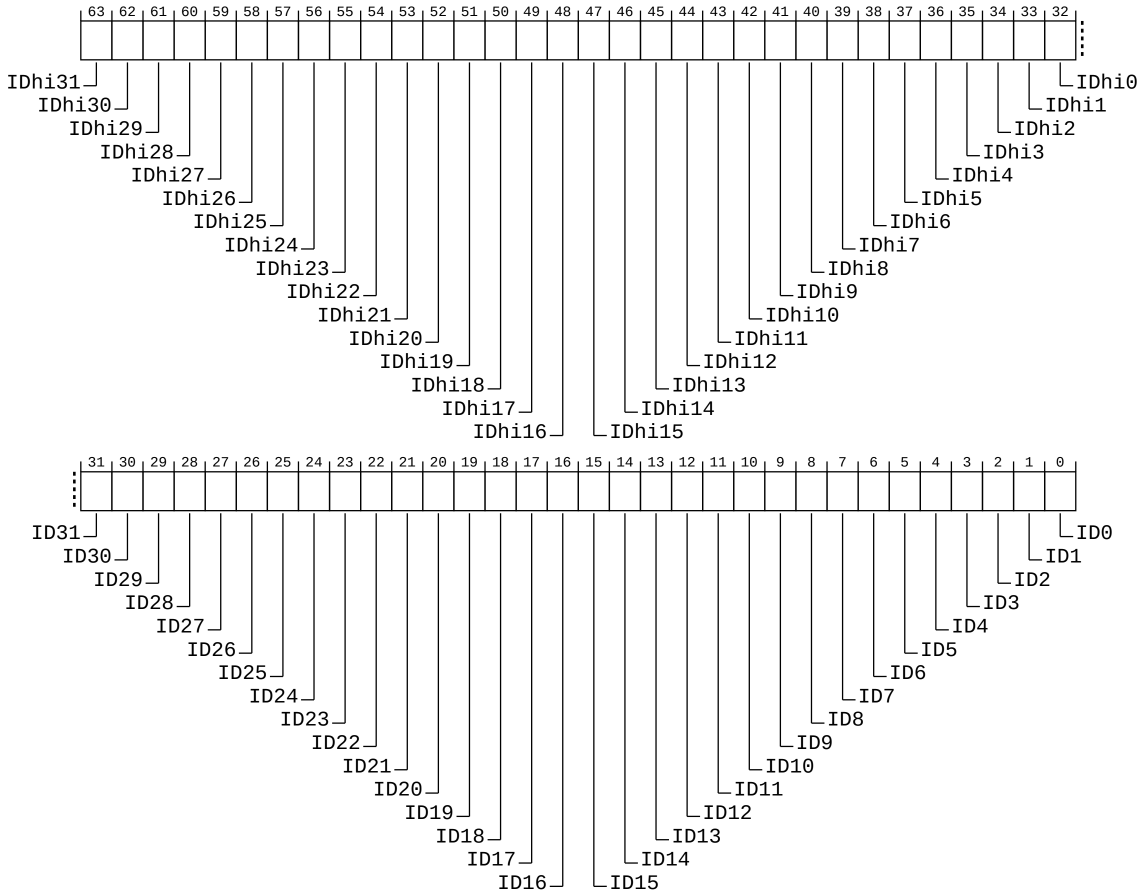


Table B-227: PMCEID0_EL0 bit descriptions

Bits	Name	Description	Reset
[63]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x401f) **0** The common event is not implemented, or not counted.	
[62]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x401e) **0** The common event is not implemented, or not counted.	
[61]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x401d) **0** The common event is not implemented, or not counted.	
[60]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x401c) **0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[59]	IDhi27	IDhi27 corresponds to common event (0x401b) CTI_TRIGOUT7 `1` The common event is implemented.	
[58]	IDhi26	IDhi26 corresponds to common event (0x401a) CTI_TRIGOUT6 `1` The common event is implemented.	
[57]	IDhi25	IDhi25 corresponds to common event (0x4019) CTI_TRIGOUT5 `1` The common event is implemented.	
[56]	IDhi24	IDhi24 corresponds to common event (0x4018) CTI_TRIGOUT4 `1` The common event is implemented.	
[55]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4017) `0` The common event is not implemented, or not counted.	
[54]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4016) `0` The common event is not implemented, or not counted.	
[53]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4015) `0` The common event is not implemented, or not counted.	
[52]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4014) `0` The common event is not implemented, or not counted.	
[51]	IDhi19	IDhi19 corresponds to common event (0x4013) TRCEXTOUT3 `1` The common event is implemented.	
[50]	IDhi18	IDhi18 corresponds to common event (0x4012) TRCEXTOUT2 `1` The common event is implemented.	
[49]	IDhi17	IDhi17 corresponds to common event (0x4011) TRCEXTOUT1 `1` The common event is implemented.	
[48]	IDhi16	IDhi16 corresponds to common event (0x4010) TRCEXTOUT0 `1` The common event is implemented.	
[47]	IDhi15	IDhi15 corresponds to common event (0x400f) PMU_HOVFS `0` The common event is not implemented, or not counted.	
[46]	IDhi14	IDhi14 corresponds to common event (0x400e) TRB_TRIG `1` The common event is implemented.	

Bits	Name	Description	Reset
[45]	IDhi13	IDhi13 corresponds to common event (0x400d) PMU_OVFS **0** The common event is not implemented, or not counted.	
[44]	IDhi12	IDhi12 corresponds to common event (0x400c) TRB_WRAP **1** The common event is implemented.	
[43]	IDhi11	IDhi11 corresponds to common event (0x400b) L3D_CACHE_LMISS_RD **0** The common event is not implemented, or not counted. This value is reported if either the Cortex®-A510 complex is configured without an L2 cache or the DSU is configured without an L3 cache. **1** The common event is implemented. This value is reported if both the Cortex®-A510 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	
[42]	IDhi10	IDhi10 corresponds to common event (0x400a) L2I_CACHE_LMISS **0** The common event is not implemented, or not counted.	
[41]	IDhi9	IDhi9 corresponds to common event (0x4009) L2D_CACHE_LMISS_RD **0** The common event is not implemented, or not counted. This value is reported if both the Cortex®-A510 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **1** The common event is implemented. This value is reported if either the Cortex®-A510 complex is configured with an L2 cache or the DSU is configured with an L3 cache.	
[40]	IDhi8	IDhi8 corresponds to common event (0x4008) Reserved **0** The common event is not implemented, or not counted.	
[39]	IDhi7	IDhi7 corresponds to common event (0x4007) Reserved **0** The common event is not implemented, or not counted.	
[38]	IDhi6	IDhi6 corresponds to common event (0x4006) L1I_CACHE_LMISS **1** The common event is implemented.	
[37]	IDhi5	IDhi5 corresponds to common event (0x4005) STALL_BACKEND_MEM **1** The common event is implemented.	
[36]	IDhi4	IDhi4 corresponds to common event (0x4004) CNT_CYCLES **0** The common event is not implemented, or not counted.	
[35]	IDhi3	IDhi3 corresponds to common event (0x4003) SAMPLE_COLLISION **0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[34]	IDhi2	IDhi2 corresponds to common event (0x4002) SAMPLE_FILTRATE **0** The common event is not implemented, or not counted.	
[33]	IDhi1	IDhi1 corresponds to common event (0x4001) SAMPLE_FEED **0** The common event is not implemented, or not counted.	
[32]	IDhi0	IDhi0 corresponds to common event (0x4000) SAMPLE_POP **0** The common event is not implemented, or not counted.	
[31]	ID31	ID31 corresponds to common event (0x1f) L1D_CACHE_ALLOCATE **0** The common event is not implemented, or not counted.	
[30]	ID30	ID30 corresponds to common event (0x1e) CHAIN **1** The common event is implemented.	
[29]	ID29	ID29 corresponds to common event (0x1d) BUS_CYCLES **1** The common event is implemented.	
[28]	ID28	ID28 corresponds to common event (0x1c) TTBR_WRITE_RETIRED **1** The common event is implemented.	
[27]	ID27	ID27 corresponds to common event (0x1b) INST_SPEC **1** The common event is implemented.	
[26]	ID26	ID26 corresponds to common event (0x1a) MEMORY_ERROR **1** The common event is implemented.	
[25]	ID25	ID25 corresponds to common event (0x19) BUS_ACCESS **1** The common event is implemented.	
[24]	ID24	ID24 corresponds to common event (0x18) L2D_CACHE_WB **0** The common event is not implemented, or not counted. This value is reported if the Cortex®-A510 complex is configured without an L2 cache. **1** The common event is implemented. This value is reported if the Cortex®-A510 complex is configured with an L2 cache.	

Bits	Name	Description	Reset
[23]	ID23	ID23 corresponds to common event (0x17) L2D_CACHE_REFILL **0** The common event is not implemented, or not counted. This value is reported if the Cortex®-A510 complex is configured without an L2 cache. **1** The common event is implemented. This value is reported if the Cortex®-A510 complex is configured with an L2 cache.	
[22]	ID22	ID22 corresponds to common event (0x16) L2D_CACHE **0** The common event is not implemented, or not counted. This value is reported if both the Cortex®-A510 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **1** The common event is implemented. This value is reported if either the Cortex®-A510 complex is configured with an L2 cache or the DSU is configured with an L3 cache.	
[21]	ID21	ID21 corresponds to common event (0x15) L1D_CACHE_WB **1** The common event is implemented.	
[20]	ID20	ID20 corresponds to common event (0x14) L1I_CACHE **1** The common event is implemented.	
[19]	ID19	ID19 corresponds to common event (0x13) MEM_ACCESS **1** The common event is implemented.	
[18]	ID18	ID18 corresponds to common event (0x12) BR_PRED **1** The common event is implemented.	
[17]	ID17	ID17 corresponds to common event (0x11) CPU_CYCLES **1** The common event is implemented.	
[16]	ID16	ID16 corresponds to common event (0x10) BR_MIS_PRED **1** The common event is implemented.	
[15]	ID15	ID15 corresponds to common event (0xf) UNALIGNED_LDST_RETIRED **0** The common event is not implemented, or not counted.	
[14]	ID14	ID14 corresponds to common event (0xe) BR_RETURN_RETIRED **1** The common event is implemented.	
[13]	ID13	ID13 corresponds to common event (0xd) BR_IMMED_RETIRED **1** The common event is implemented.	

Bits	Name	Description	Reset
[12]	ID12	ID12 corresponds to common event (0xc) PC_WRITE_RETIRED **1** The common event is implemented.	
[11]	ID11	ID11 corresponds to common event (0xb) CID_WRITE_RETIRED **1** The common event is implemented.	
[10]	ID10	ID10 corresponds to common event (0xa) EXC_RETURN **1** The common event is implemented.	
[9]	ID9	ID9 corresponds to common event (0x9) EXC_TAKEN **1** The common event is implemented.	
[8]	ID8	ID8 corresponds to common event (0x8) INST_RETIRED **1** The common event is implemented.	
[7]	ID7	ID7 corresponds to common event (0x7) ST_RETIRED **1** The common event is implemented.	
[6]	ID6	ID6 corresponds to common event (0x6) LD_RETIRED **1** The common event is implemented.	
[5]	ID5	ID5 corresponds to common event (0x5) L1D_TLB_REFILL **1** The common event is implemented.	
[4]	ID4	ID4 corresponds to common event (0x4) L1D_CACHE **1** The common event is implemented.	
[3]	ID3	ID3 corresponds to common event (0x3) L1D_CACHE_REFILL **1** The common event is implemented.	
[2]	ID2	ID2 corresponds to common event (0x2) L1I_TLB_REFILL **1** The common event is implemented.	
[1]	ID1	ID1 corresponds to common event (0x1) L1I_CACHE_REFILL **1** The common event is implemented.	
[0]	ID0	ID0 corresponds to common event (0x0) SW_INCR **1** The common event is implemented.	

Access

MRS <Xt>, PMCEID0_EL0

<systemreg>	op0	op1	CRn	CRm	op2
PMCEID0_EL0	0b11	0b011	0b1001	0b1100	0b110

Accessibility

MRS <Xt>, PMCEID0_EL0

```
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_EL0;
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_EL0;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID0_EL0;
elsif PSTATE.EL == EL3 then
    return PMCEID0_EL0;
```

B.6.4 PMCEID1_EL0, 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.

Note

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

For more information about the common events and the use of the PMCEID<n>_EL0 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 B-96: AArch64_pmceid1_el0 bit assignments

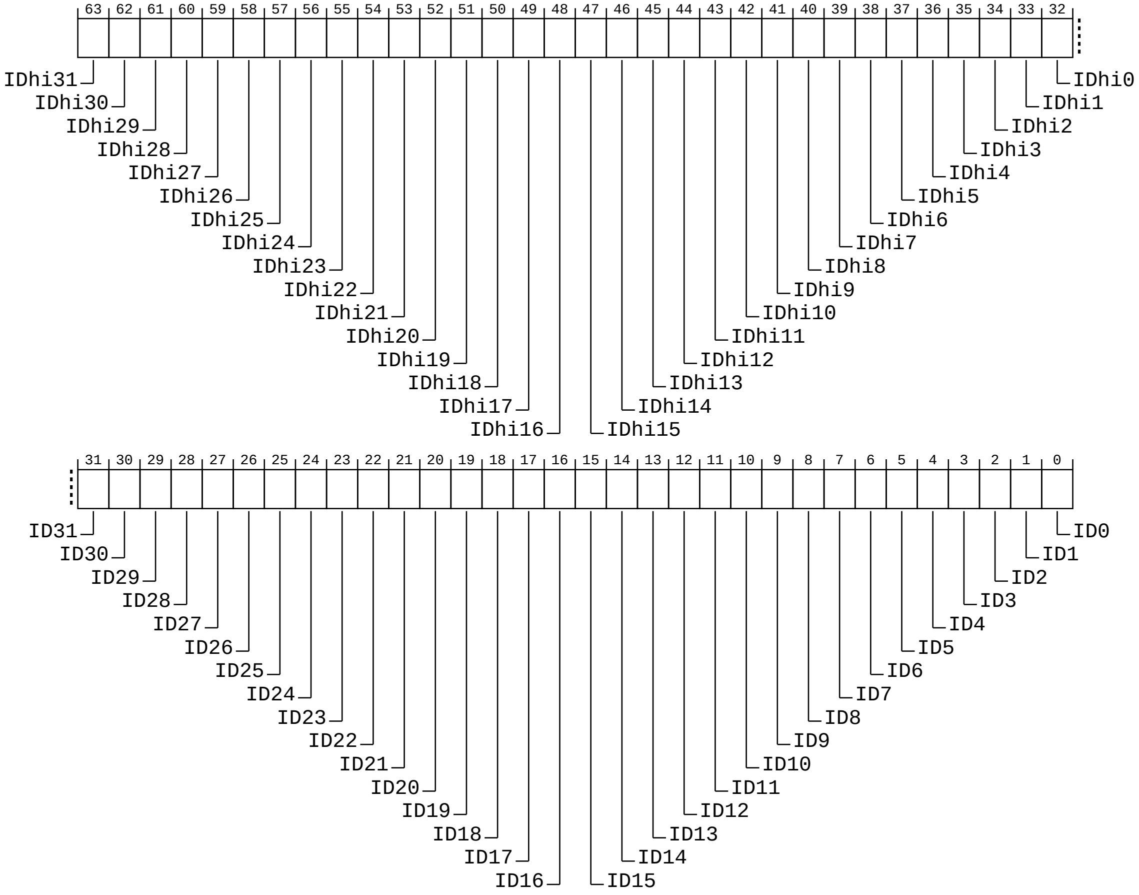


Table B-229: PMCEID1_EL0 bit descriptions

Bits	Name	Description	Reset
[63]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x403f) **0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[62]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x403e) **0** The common event is not implemented, or not counted.	
[61]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x403d) **0** The common event is not implemented, or not counted.	
[60]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x403c) **0** The common event is not implemented, or not counted.	
[59]	IDhi27	IDhi27 corresponds to a Reserved Event event (0x403b) **0** The common event is not implemented, or not counted.	
[58]	IDhi26	IDhi26 corresponds to a Reserved Event event (0x403a) **0** The common event is not implemented, or not counted.	
[57]	IDhi25	IDhi25 corresponds to a Reserved Event event (0x4039) **0** The common event is not implemented, or not counted.	
[56]	IDhi24	IDhi24 corresponds to a Reserved Event event (0x4038) **0** The common event is not implemented, or not counted.	
[55]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4037) **0** The common event is not implemented, or not counted.	
[54]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4036) **0** The common event is not implemented, or not counted.	
[53]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4035) **0** The common event is not implemented, or not counted.	
[52]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4034) **0** The common event is not implemented, or not counted.	
[51]	IDhi19	IDhi19 corresponds to a Reserved Event event (0x4033) **0** The common event is not implemented, or not counted.	
[50]	IDhi18	IDhi18 corresponds to a Reserved Event event (0x4032) **0** The common event is not implemented, or not counted.	
[49]	IDhi17	IDhi17 corresponds to a Reserved Event event (0x4031) **0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[48]	IDhi16	IDhi16 corresponds to a Reserved Event event (0x4030) **0** The common event is not implemented, or not counted.	
[47]	IDhi15	IDhi15 corresponds to a Reserved Event event (0x402f) **0** The common event is not implemented, or not counted.	
[46]	IDhi14	IDhi14 corresponds to a Reserved Event event (0x402e) **0** The common event is not implemented, or not counted.	
[45]	IDhi13	IDhi13 corresponds to a Reserved Event event (0x402d) **0** The common event is not implemented, or not counted.	
[44]	IDhi12	IDhi12 corresponds to a Reserved Event event (0x402c) **0** The common event is not implemented, or not counted.	
[43]	IDhi11	IDhi11 corresponds to a Reserved Event event (0x402b) **0** The common event is not implemented, or not counted.	
[42]	IDhi10	IDhi10 corresponds to a Reserved Event event (0x402a) **0** The common event is not implemented, or not counted.	
[41]	IDhi9	IDhi9 corresponds to a Reserved Event event (0x4029) **0** The common event is not implemented, or not counted.	
[40]	IDhi8	IDhi8 corresponds to a Reserved Event event (0x4028) **0** The common event is not implemented, or not counted.	
[39]	IDhi7	IDhi7 corresponds to a Reserved Event event (0x4027) **0** The common event is not implemented, or not counted.	
[38]	IDhi6	IDhi6 corresponds to common event (0x4026) MEM_ACCESS_CHECKED_WR **1** The common event is implemented.	
[37]	IDhi5	IDhi5 corresponds to common event (0x4025) MEM_ACCESS_CHECKED_RD **1** The common event is implemented.	
[36]	IDhi4	IDhi4 corresponds to common event (0x4024) MEM_ACCESS_CHECKED **1** The common event is implemented.	
[35]	IDhi3	IDhi3 corresponds to common event (0x4023) Reserved **0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[34]	IDhi2	IDhi2 corresponds to common event (0x4022) ST_ALIGN_LAT **1** The common event is implemented.	
[33]	IDhi1	IDhi1 corresponds to common event (0x4021) LD_ALIGN_LAT **1** The common event is implemented.	
[32]	IDhi0	IDhi0 corresponds to common event (0x4020) LDST_ALIGN_LAT **1** The common event is implemented.	
[31]	ID31	ID31 corresponds to common event (0x3f) STALL_SLOT **1** The common event is implemented.	
[30]	ID30	ID30 corresponds to common event (0x3e) STALL_SLOT_FRONTEND **1** The common event is implemented.	
[29]	ID29	ID29 corresponds to common event (0x3d) STALL_SLOT_BACKEND **1** The common event is implemented.	
[28]	ID28	ID28 corresponds to common event (0x3c) STALL **1** The common event is implemented.	
[27]	ID27	ID27 corresponds to common event (0x3b) OP_SPEC **1** The common event is implemented.	
[26]	ID26	ID26 corresponds to common event (0x3a) OP_RETIRED **1** The common event is implemented.	
[25]	ID25	ID25 corresponds to common event (0x39) L1D_CACHE_LMISS_RD **1** The common event is implemented.	
[24]	ID24	ID24 corresponds to common event (0x38) REMOTE_ACCESS_RD **1** The common event is implemented.	
[23]	ID23	ID23 corresponds to common event (0x37) LL_CACHE_MISS_RD **1** The common event is implemented.	
[22]	ID22	ID22 corresponds to common event (0x36) LL_CACHE_RD **1** The common event is implemented.	
[21]	ID21	ID21 corresponds to common event (0x35) ITLB_WLK **1** The common event is implemented.	

Bits	Name	Description	Reset
[20]	ID20	ID20 corresponds to common event (0x34) DTLB_WLK **1** The common event is implemented.	
[19]	ID19	ID19 corresponds to a Reserved Event event (0x33) **0** The common event is not implemented, or not counted.	
[18]	ID18	ID18 corresponds to a Reserved Event event (0x32) **0** The common event is not implemented, or not counted.	
[17]	ID17	ID17 corresponds to common event (0x31) REMOTE_ACCESS **0** The common event is not implemented, or not counted.	
[16]	ID16	ID16 corresponds to common event (0x30) L2I_TLB **0** The common event is not implemented, or not counted.	
[15]	ID15	ID15 corresponds to common event (0x2f) L2D_TLB **1** The common event is implemented.	
[14]	ID14	ID14 corresponds to common event (0x2e) L2I_TLB_REFILL **0** The common event is not implemented, or not counted.	
[13]	ID13	ID13 corresponds to common event (0x2d) L2D_TLB_REFILL **1** The common event is implemented.	
[12]	ID12	ID12 corresponds to common event (0x2c) Reserved **0** The common event is not implemented, or not counted.	
[11]	ID11	ID11 corresponds to common event (0x2b) L3D_CACHE **0** The common event is not implemented, or not counted. This value is reported if either the Cortex®-A510 complex is configured without an L2 cache or the DSU is configured without an L3 cache. **1** The common event is implemented. This value is reported if both the Cortex®-A510 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	
[10]	ID10	ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL **0** The common event is not implemented, or not counted.	
[9]	ID9	ID9 corresponds to common event (0x29) L3D_CACHE_ALLOCATE **0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[8]	ID8	ID8 corresponds to common event (0x28) L2I_CACHE_REFILL **0** The common event is not implemented, or not counted.	
[7]	ID7	ID7 corresponds to common event (0x27) L2I_CACHE **0** The common event is not implemented, or not counted.	
[6]	ID6	ID6 corresponds to common event (0x26) L1I_TLB **1** The common event is implemented.	
[5]	ID5	ID5 corresponds to common event (0x25) L1D_TLB **1** The common event is implemented.	
[4]	ID4	ID4 corresponds to common event (0x24) STALL_BACKEND **1** The common event is implemented.	
[3]	ID3	ID3 corresponds to common event (0x23) STALL_FRONTEND **1** The common event is implemented.	
[2]	ID2	ID2 corresponds to common event (0x22) BR_MIS_PRED_RETIRED **1** The common event is implemented.	
[1]	ID1	ID1 corresponds to common event (0x21) BR_RETIRED **1** The common event is implemented.	
[0]	ID0	ID0 corresponds to common event (0x20) L2D_CACHE_ALLOCATE **0** The common event is not implemented, or not counted. This value is reported if the Cortex®-A510 complex is configured without an L2 cache. **1** The common event is implemented. This value is reported if the Cortex®-A510 complex is configured with an L2 cache.	

Access

MRS <Xt>, PMCEID1_EL0

<systemreg>	op0	op1	CRn	CRm	op2
PMCEID1_EL0	`0b11`	`0b011`	`0b1001`	`0b1100`	`0b111`

Accessibility

MRS <Xt>, PMCEID1_EL0

```
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_EL0;
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_EL0;
elsif PSTATE.EL == EL2 then
    if MDCR_EL3.TPM == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return PMCEID1_EL0;
elsif PSTATE.EL == EL3 then
    return PMCEID1_EL0;
```

B.7 GIC register summary

The summary table provides an overview of *Generic Interrupt Controller* (GIC) registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table B-231: 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)

B.7.1 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 B-97: AArch64_icc_ctlr_el1 bit assignments

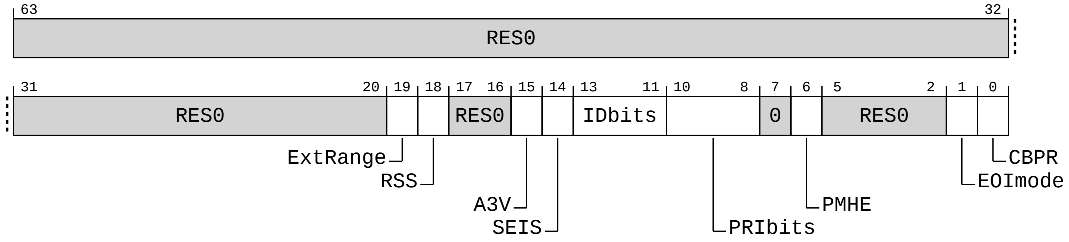


Table B-232: ICC_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	`0b0`
[19]	ExtRange	Extended INTID range (read-only). **1** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	
[18]	RSS	Range Selector Support. Possible values are: **0** Targeted SGIs with affinity level 0 values of 0 - 15 are supported.	
[17:16]	RES0	Reserved	`0b0`
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. Possible values are: **1** The CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.	

Bits	Name	Description	Reset
[14]	SEIS	SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs: **0** 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: **000** 16 bits.	
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits). An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits). **Note:** This field always returns the number of priority bits implemented, regardless of the Security state of the access or the value of ext-GICD_CTLR.DS. For physical accesses, this field determines the minimum value of AArch64-ICC_BPR0_EL1. If EL3 is implemented, physical accesses return the value from AArch64-ICC_CTLR_EL3.PRIbits. **100** 5 bits of priority are implemented	
[7]	RES0	Reserved	0b0
[6]	PMHE	Priority Mask Hint Enable. Controls whether the priority mask register is used as a hint for interrupt distribution: **0** Disables use of AArch64-ICC_PMR_EL1 as a hint for interrupt distribution. **1** Enables use of AArch64-ICC_PMR_EL1 as a hint for interrupt distribution. 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: • If ext-GICD_CTLR.DS == 0, this bit is read-only. • If ext-GICD_CTLR.DS == 1, this bit is read/write.	
[5:2]	RES0	Reserved	0b0
[1]	EOImode	EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt: **0** 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. **1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality. The Secure AArch64-ICC_CTLR_EL1.EOImode is an alias of AArch64-ICC_CTLR_EL3.EOImode_EL1S. The Non-secure AArch64-ICC_CTLR_EL1.EOImode is an alias of AArch64-ICC_CTLR_EL3.EOImode_EL1NS	

Bits	Name	Description	Reset
[0]	CBPR	Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupts: **0** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. AArch64-ICC_BPR1_EL1 determines the preemption group for Group 1 interrupts. **1** AArch64-ICC_BPR0_EL1 determines the preemption group for both Group 0 and Group 1 interrupts. If EL3 is implemented: • This bit is an alias of AArch64-ICC_CTLR_EL3.CBPR_EL1{S,NS} where S or NS corresponds to the current Security state. • If ext-GICD_CTLR.DS == 0, this bit is read-only. • If ext-GICD_CTLR.DS == 1, this bit is read/write.	

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];
```

B.7.2 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 B-98: AArch64_icv_ctlr_el1 bit assignments

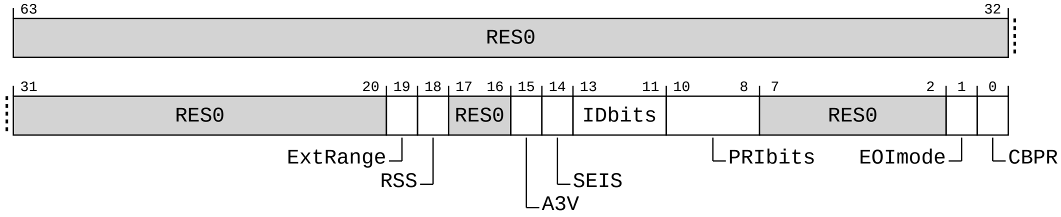


Table B-235: ICV_CTLR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	`0b0`
[19]	ExtRange	Extended INTID range (read-only). **`1`** CPU interface supports INTIDs in the range 1024..8191 - All INTIDs in the range 1024..8191 are treated as requiring deactivation.	
[18]	RSS	Range Selector Support. Possible values are: **`0`** Targeted SGIs with affinity level 0 values of 0 - 15 are supported.	
[17:16]	RES0	Reserved	`0b0`
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. Possible values are: **`1`** 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: **`0`** 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: **`000`** 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). **Note:** This field always returns the number of priority bits implemented. The division between group priority and subpriority is defined in the binary point registers AArch64-ICV_BPR0_EL1 and AArch64-ICV_BPR1_EL1. **`100`** 5 bits of priority are implemented	
[7:2]	RES0	Reserved	`0b0`

Bits	Name	Description	Reset
[1]	EOImode	Virtual EOI mode. Controls whether a write to an End of Interrupt register also deactivates the virtual interrupt: **0** 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. **1** AArch64-ICV_EOIR0_EL1 and AArch64-ICV_EOIR1_EL1 provide priority drop functionality only. AArch64-ICV_DIR_EL1 provides interrupt deactivation functionality.	
[0]	CBPR	Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both virtual Group 0 and virtual Group 1 interrupts: **0** AArch64-ICV_BPR1_EL1 determines the preemption group for virtual Group 1 interrupts. **1** Reads of AArch64-ICV_BPR1_EL1 return AArch64-ICV_BPR0_EL1 plus one, saturated to 0b111. Writes to AArch64-ICV_BPR1_EL1 are ignored.	

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];
```

B.7.3 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

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Figure B-99: AArch64_icc_ap0r0_el1 bit assignments

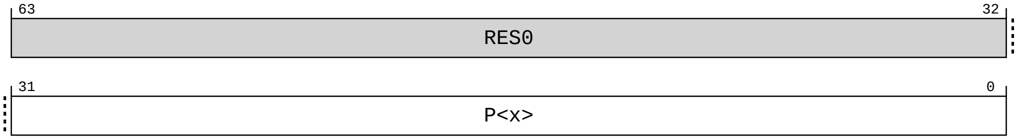


Table B-238: ICC_AP0R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:0]	P<x>	Provides the access to the active priorities for Group 0 interrupts. Possible values of each bit are: **0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	

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

B.7.4 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

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Figure B-100: AArch64_icv_ap0r0_el1 bit assignments

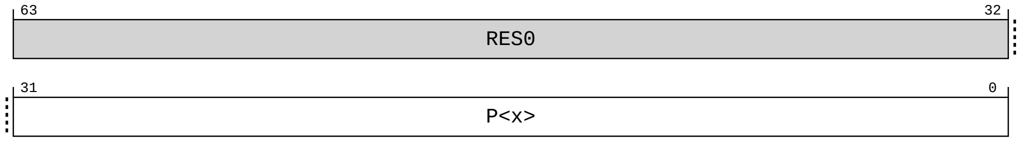


Table B-241: ICV_AP0R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:0]	P<x>	Provides the access to the virtual active priorities for Group 0 interrupts. Possible values of each bit are: **0** There is no Group 0 interrupt active with this priority level, or all active Group 0 interrupts with this priority level have undergone priority-drop. **1** There is a Group 0 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	

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

B.7.5 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

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Figure B-101: AArch64_icc_ap1r0_el1 bit assignments

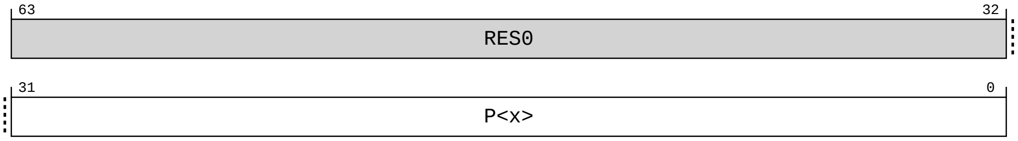


Table B-244: ICC_AP1R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0b0`
[31:0]	P<x>	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: **0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3]. When accessed from non-secure EL2 or EL1, only the 16 lowest-priority interrupts are visible in bits [15:0] of this register.	

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

B.7.6 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

The contents of these registers are **IMPLEMENTATION DEFINED** with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Figure B-102: AArch64_icv_ap1r0_el1 bit assignments

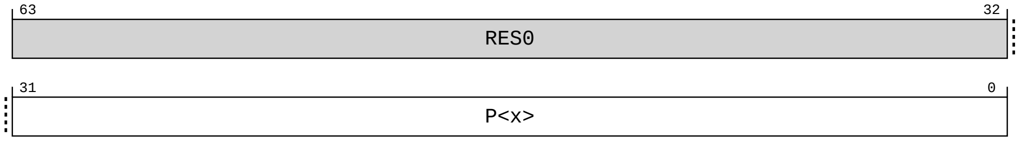


Table B-247: ICV_AP1R0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[31:0]	P<x>	Group 1 interrupt active priorities. Possible values of each bit are: **0** There is no Group 1 interrupt active with this priority level, or all active Group 1 interrupts with this priority level have undergone priority-drop. **1** There is a Group 1 interrupt active with this priority level which has not undergone priority drop. There are 32 preemption levels, and the active state of these preemption levels are held in the bits corresponding to Priority[7:3].	

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`

B.7.7 ICH_VTR_EL2, Interrupt Controller VGIC Type Register

Reports supported GIC virtualization features.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

GIC

Reset value

See individual bit resets.

Bit descriptions

Figure B-103: AArch64_ich_vtr_el2 bit assignments

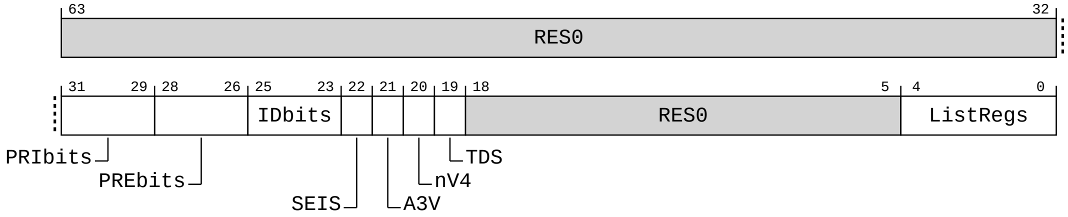


Table B-250: ICH_VTR_EL2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0b0`
[31:29]	PRIbits	Priority bits. The number of virtual priority bits implemented, minus one. An implementation must implement at least 32 levels of virtual priority (5 priority bits). This field is an alias of AArch64-ICV_CTLR_EL1.PRIbits. **100** 5 virtual priority bits are implemented	
[28:26]	PREbits	The number of virtual preemption bits implemented, minus one. An implementation must implement at least 32 levels of virtual preemption priority (5 preemption bits). The value of this field must be less than or equal to the value of ICH_VTR_EL2.PRIbits. The maximum value of this field is 6, indicating 7 bits of preemption. This field determines the minimum value of AArch64-ICH_VMCR_EL2.VBPR0. **100** 5 virtual pre-emption bits are implemented	
[25:23]	IDbits	The number of virtual interrupt identifier bits supported: **000** 16 bits.	
[22]	SEIS	SEI Support. Indicates whether the virtual CPU interface supports generation of SEIs: **0** The virtual CPU interface logic does not support generation of SEIs.	
[21]	A3V	Affinity 3 Valid. Possible values are: **1** The virtual CPU interface logic supports non-zero values of Affinity 3 in SGI generation System registers.	
[20]	nV4	Direct injection of virtual interrupts not supported. Possible values are: **0** The CPU interface logic supports direct injection of virtual interrupts.	

Bits	Name	Description	Reset
[19]	TDS	Separate trapping of EL1 writes to AArch64-ICV_DIR_EL1 supported. **1** Implementation supports AArch64-ICH_HCR_EL2.TDIR.	
[18:5]	RES0	Reserved	0b0
[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. **00011** Four list registers are implemented.	

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;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    return ICH_VTR_EL2;
elsif PSTATE.EL == EL3 then
    return ICH_VTR_EL2;
```

B.7.8 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 B-104: AArch64_icc_ctlr_el3 bit assignments

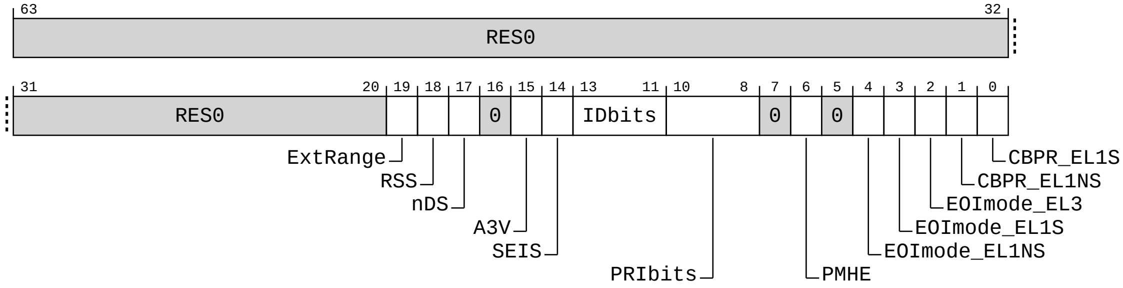


Table B-252: ICC_CTLR_EL3 bit descriptions

Bits	Name	Description	Reset
[63:20]	RES0	Reserved	0b0
[19]	ExtRange	Extended INTID range (read-only). **1** CPU interface supports INTIDs in the range 1024..8191 • All INTIDs in the range 1024..8191 are treated as requiring deactivation.	
[18]	RSS	Range Selector Support. **0** Targeted SGIs with affinity level 0 values of 0-15 are supported.	
[17]	nDS	Disable Security not supported. Read-only and writes are ignored. **1** The CPU interface logic does not support disabling of security, and requires that security is not disabled.	
[16]	RES0	Reserved	0b0
[15]	A3V	Affinity 3 Valid. Read-only and writes are ignored. **1** 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: **0** 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. **000** 16 bits.	

Bits	Name	Description	Reset
[10:8]	PRIbits	Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one. An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits). An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits). **Note:** This field always returns the number of priority bits implemented, regardless of the value of SCR_EL3.NS or the value of 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. This field determines the minimum value of ICC_BPR0_EL1. **100** 5 bits of priority are implemented	
[7]	RES0	Reserved	0b0
[6]	PMHE	Priority Mask Hint Enable. **0** Disables use of the priority mask register as a hint for interrupt distribution. **1** Enables use of the priority mask register as a hint for interrupt distribution. Software must write AArch64-ICC_PMR_EL1 to 0xFF before clearing this field to 0. • An implementation might choose to make this field RAO/WI if priority-based routing is always used • An implementation might choose to make this field RAZ/WI if priority-based routing is never used If EL3 is present, AArch64-ICC_CTLR_EL1.PMHE is an alias of ICC_CTLR_EL3.PMHE.	0b0
[5]	RES0	Reserved	0b0
[4]	EOImode_EL1NS	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. **0** 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. **1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality. If EL3 is present, AArch64-ICC_CTLR_EL1(NS).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1NS.	

Bits	Name	Description	Reset
[3]	EOImode_EL1S	EOI mode for interrupts handled at Secure EL1. Controls whether a write to an End of Interrupt register also deactivates the interrupt. **0** 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. **1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality. If EL3 is present, AArch64-ICC_CTLR_EL1(S).EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1S.	
[2]	EOImode_EL3	EOI mode for interrupts handled at EL3. Controls whether a write to an End of Interrupt register also deactivates the interrupt. **0** 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. **1** AArch64-ICC_EOIR0_EL1 and AArch64-ICC_EOIR1_EL1 provide priority drop functionality only. AArch64-ICC_DIR_EL1 provides interrupt deactivation functionality.	
[1]	CBPR_EL1NS	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. **0** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. AArch64-ICC_BPR1_EL1 determines the preemption group for Non-secure Group 1 interrupts. **1** 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. If EL3 is present, AArch64-ICC_CTLR_EL1(NS).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1NS.	
[0]	CBPR_EL1S	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. **0** AArch64-ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. AArch64-ICC_BPR1_EL1 determines the preemption group for Secure Group 1 interrupts. **1** 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. If EL3 is present, AArch64-ICC_CTLR_EL1(S).CBPR is an alias of ICC_CTLR_EL3.CBPR_EL1S.	

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;
elsif PSTATE.EL == EL1 then
    UNDEFINED;
elsif PSTATE.EL == EL2 then
    UNDEFINED;
elsif PSTATE.EL == EL3 then
    return ICC_CTLR_EL3;
```

MSR ICC_CTLR_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
    ICC_CTLR_EL3 = X[t];
```

B.8 Activity monitors register summary

The summary table provides an overview of activity monitors registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table B-255: Activity monitors register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
AMEVTYPER10_EL0	3	C13	3	C14	0	See individual bit resets	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER11_EL0	3	C13	3	C14	1	See individual bit resets	64-bit	Activity Monitors Event Type Registers 1
AMEVTYPER12_EL0	3	C13	3	C14	2	See individual bit resets	64-bit	Activity Monitors Event Type Registers 1
AMCFGR_EL0	3	C13	3	C2	1	See individual bit resets	64-bit	Activity Monitors Configuration Register
AMCGCR_EL0	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_EL0	3	C13	3	C6	0	See individual bit resets	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER01_EL0	3	C13	3	C6	1	See individual bit resets	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER02_EL0	3	C13	3	C6	2	See individual bit resets	64-bit	Activity Monitors Event Type Registers 0
AMEVTYPER03_EL0	3	C13	3	C6	3	See individual bit resets	64-bit	Activity Monitors Event Type Registers 0

B.8.1 AMEVTYPER10_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AArch64-AMEVCNTR10_EL0 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 B-105: AArch64_amevtyper10_el0 bit assignments

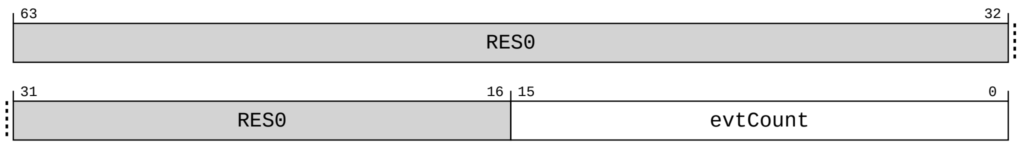


Table B-256: AMEVTYPER10_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	`0b0`
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AMEVCNTR10_EL0. **0000001100000000** MPMM gear 0 period threshold exceeded	

Access

MRS <Xt>, AMEVTYPER10_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER10_EL0	0b11	0b011	0b1101	0b1110	0b000

B.8.2 AMEVTYPER11_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AArch64-AMEVCNTR11_EL0 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 B-106: AArch64_amevtyper11_el0 bit assignments

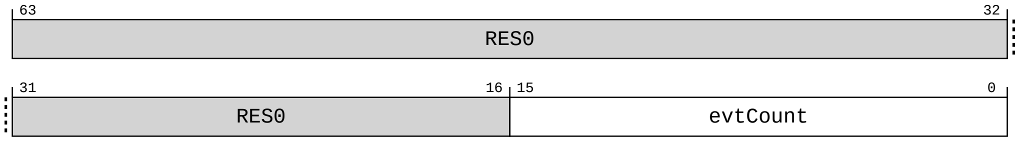


Table B-258: AMEVTYPER11_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0b0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AMEVCNTR11_EL0. **0000001100000001** MPMM gear 1 period threshold exceeded	

Access

MRS <Xt>, AMEVTYPER11_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER11_EL0	0b11	0b011	0b1101	0b1110	0b001

B.8.3 AMEVTYPER12_EL0, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AArch64-AMEVCNTR12_EL0 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 B-107: AArch64_amevtyper12_el0 bit assignments

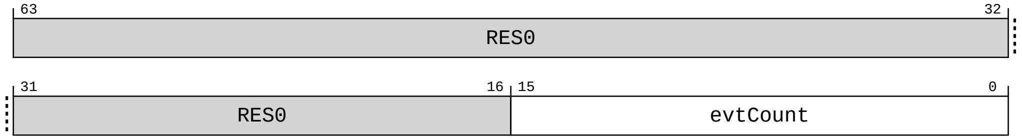


Table B-260: AMEVTYPER12_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0b0
[15:0]	evtCount	Event to count. The event number of the event that is counted by the auxiliary activity monitor event counter AMEVCNTR12_EL0. **0000001100000010** MPMM gear 2 period threshold exceeded	

Access

MRS <Xt>, AMEVTYPER12_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER12_EL0	0b11	0b011	0b1101	0b1110	0b010

B.8.4 AMCFGR_EL0, 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_EL0 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 B-108: AArch64_amcfgr_el0 bit assignments

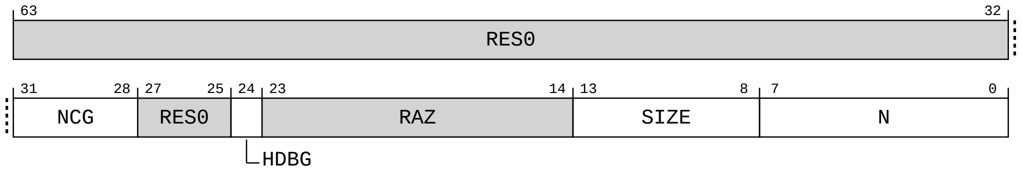


Table B-262: AMCFGR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0b0`
[31:28]	NCG	Defines the number of counter groups. The following value is specified for this product. **`0001`** 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. **`1`** AArch64-AMCR_EL0.HDBG is read/write.	
[23:14]	RAZ	Reserved	

Bits	Name	Description	Reset
[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_EL0.SIZE + 1]. From Armv8, the counters are 64-bit, and so this field is 0b111111. **Note:** Software also uses this field to determine the spacing of counters in the memory-map. From Armv8, the counters are at doubleword-aligned addresses. **111111** 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_EL0.N + 1]. **00000110** Seven activity monitor event counters	

Access

MRS <Xt>, AMCFGR_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMCFGR_EL0	0b11	0b011	0b1101	0b0010	0b001

Accessibility

MRS <Xt>, AMCFGR_EL0

```
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;
```

B.8.5 AMCGCR_EL0, 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 B-109: AArch64_amcgcr_el0 bit assignments

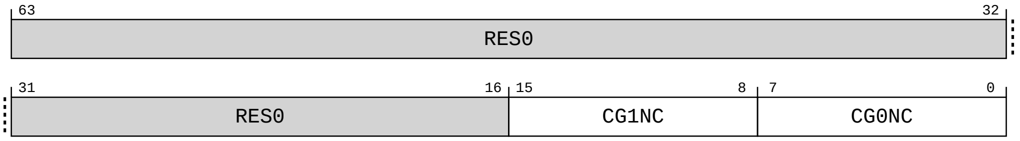


Table B-264: AMCGCR_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	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 0x0 to 0x10. **`00000011`** 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. **`00000100`** Four counters in the architected counter group	

Access

MRS <Xt>, AMCGCR_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMCGCR_EL0	0b11	0b011	0b1101	0b0010	0b010

Accessibility

MRS <Xt>, AMCGCR_EL0

```
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 AMCGCR_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 AMCGCR_EL0;
elsif PSTATE.EL == EL2 then
    if CPTR_EL3.TAM == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return AMCGCR_EL0;
elsif PSTATE.EL == EL3 then
    return AMCGCR_EL0;
```

B.8.6 AMEVTYPER00_EL0, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR00_EL0 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 B-110: AArch64_amevtyper00_el0 bit assignments

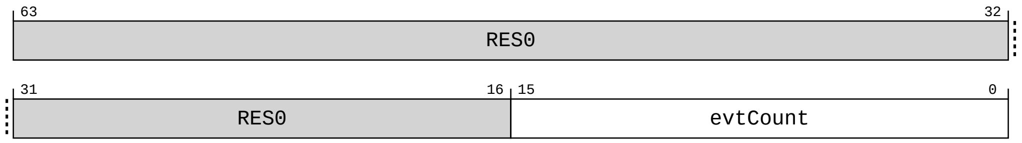


Table B-266: AMEVTYPER00_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	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 AArch64-AMEVCNTR0<n>_EL0. The value of this field is architecturally mandated for each architected counter. **0000000000010001** Processor frequency cycles	

Access

MRS <Xt>, AMEVTYPER00_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER00_EL0	0b11	0b011	0b1101	0b0110	0b000

B.8.7 AMEVTYPER01_EL0, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR01_EL0 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 B-111: AArch64_amevtyper01_el0 bit assignments

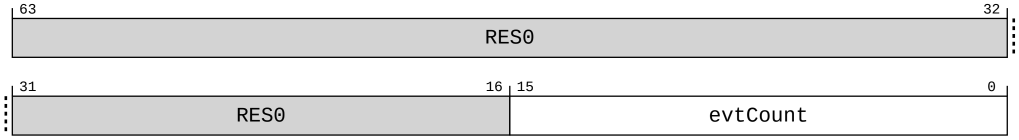


Table B-268: AMEVTYPER01_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	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 AArch64-AMEVCNTR0<n>_EL0. The value of this field is architecturally mandated for each architected counter. **`0100000000000100`** Constant frequency cycles	

Access

MRS <Xt>, AMEVTYPER01_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER01_EL0	`0b11`	`0b011`	`0b1101`	`0b0110`	`0b001`

B.8.8 AMEVTYPER02_EL0, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR02_EL0 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 B-112: AArch64_amevtyper02_el0 bit assignments

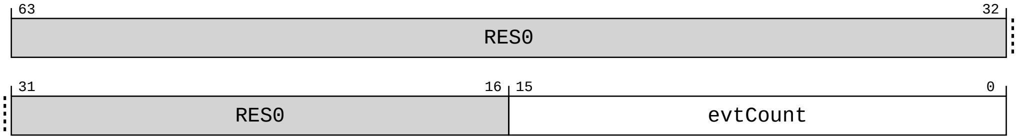


Table B-270: AMEVTYPER02_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	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 AArch64-AMEVCNTR0<n>_EL0. The value of this field is architecturally mandated for each architected counter. 0000000000001000 Instructions retired	

Access

MRS <Xt>, AMEVTYPER02_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER02_EL0	0b11	0b011	0b1101	0b0110	0b010

B.8.9 AMEVTYPER03_EL0, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR03_EL0 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 B-113: AArch64_amevtyper03_el0 bit assignments

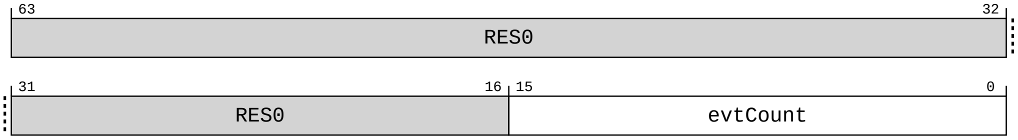


Table B-272: AMEVTYPER03_EL0 bit descriptions

Bits	Name	Description	Reset
[63:16]	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 AArch64-AMEVCNTR0<n>_EL0. The value of this field is architecturally mandated for each architected counter. **`0100000000000101`** Memory stall cycles	

Access

MRS <Xt>, AMEVTYPER03_EL0

<systemreg>	op0	op1	CRn	CRm	op2
AMEVTYPER03_EL0	`0b11`	`0b011`	`0b1101`	`0b0110`	`0b011`

B.9 RAS register summary

The summary table provides an overview of *Reliability, Availability, Serviceability* (RAS) registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table B-274: 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

B.9.1 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 B-114: AArch64_erridr_el1 bit assignments

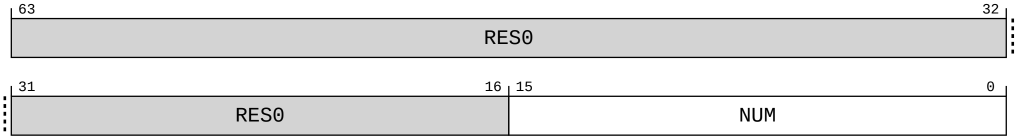


Table B-275: ERRIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	0b0
[15:0]	NUM	Highest numbered index of the records that can be accessed through the Error Record System registers plus one. Zero indicates no records can be accessed through the Error Record System registers. Each implemented record is owned by a node. A node might own multiple records. **0000000000000011** Three Records Present.	

Access

MRS <Xt>, ERRIDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ERRIDR_EL1	0b11	0b000	0b0101	0b0011	0b000

Accessibility

MRS <Xt>, 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 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;
```

B.9.2 ERRSELR_EL1, Error Record Select Register

Selects an error record to 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 B-115: AArch64_errselr_el1 bit assignments

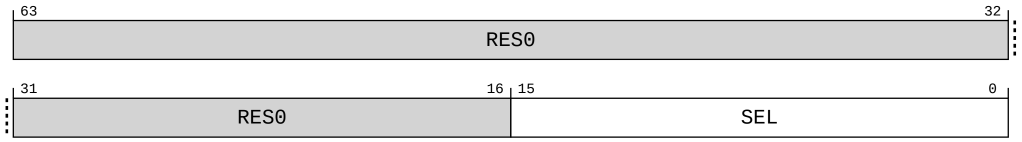


Table B-277: ERRSELR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:16]	RES0	Reserved	`0b0`
[15:0]	SEL	Selects the error record accessed through the ERX registers. **0000000000000000** Selects record 0, containing errors from DSU RAMs **0000000000000001** Selects record 1, containing errors from L1 RAMs **0000000000000010** Selects record 2, containing errors from L2 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 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 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];
```

B.10 Trace register summary

The summary table provides an overview of trace registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table B-280: 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
TRCIMSPEC0	2	C0	1	C0	7	See individual bit resets.	64-bit	IMP DEF Register 0
TRCIDR9	2	C0	1	C1	6	0x0	64-bit	ID Register 9
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
TRCIDR6	2	C0	1	C14	7	0x0	64-bit	ID Register 6
TRCIDR7	2	C0	1	C15	7	0x0	64-bit	ID Register 7
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
TRCAUXCTLR	2	C0	1	C6	0	0x0	64-bit	Auxiliary Control Register
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
TRCDEVARCH	2	C7	1	C15	6	See individual bit resets.	64-bit	Device Architecture Register
TRCDEVID	2	C7	1	C2	7	0x0	64-bit	Device Configuration Register
TRCCLAIMSET	2	C7	1	C8	6	See individual bit resets.	64-bit	Claim Tag Set Register
TRCCLAIMCLR	2	C7	1	C9	6	See individual bit resets.	64-bit	Claim Tag Clear Register

B.10.1 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 B-116: AArch64_trcidr8 bit assignments

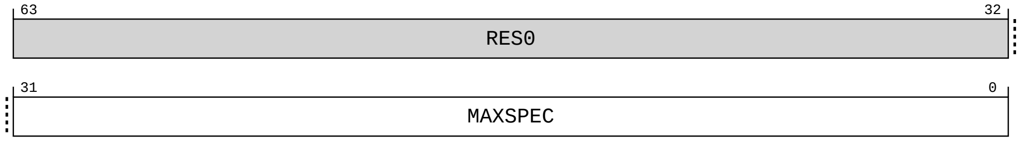


Table B-281: TRCIDR8 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:0]	MAXSPEC	Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of P0 elements in the trace element stream that can be speculative at any time. 00000000000000000000000000000000	

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;
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 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;
```

B.10.2 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

64

Functional group

Trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-117: AArch64_trcimspec0 bit assignments

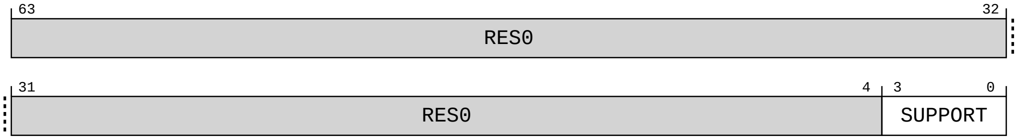


Table B-283: TRCIMSPEC0 bit descriptions

Bits	Name	Description	Reset
[63:4]	RES0	Reserved	0b0
[3:0]	SUPPORT	Indicates whether the implementation supports **IMPLEMENTATION DEFINED** features. **0000** No **IMPLEMENTATION DEFINED** features are supported.	

Access

MRS <Xt>, TRCIMSPEC0

<systemreg>	op0	op1	CRn	CRm	op2
TRCIMSPEC0	0b10	0b001	0b0000	0b0000	0b111

MSR TRCIMSPEC0, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
TRCIMSPEC0	0b10	0b001	0b0000	0b0000	0b111

Accessibility

MRS <Xt>, TRCIMSPEC0

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPEC0;
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 TRCIMSPEC0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIMSPEC0;
```

MSR TRCIMSPEC0, <Xt>

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t];
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
        TRCIMSPEC0 = X[t];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCIMSPEC0 = X[t];
```

B.10.3 TRCIDR9, ID Register 9

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 B-118: AArch64_trcidr9 bit assignments

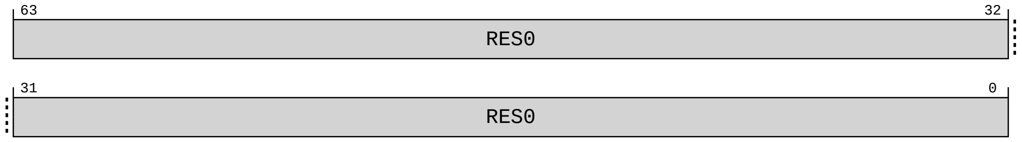


Table B-286: TRCIDR9 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

Access

MRS <Xt>, TRCIDR9

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR9	0b10	0b001	0b0000	0b0001	0b110

Accessibility

MRS <Xt>, TRCIDR9

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR9;
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 TRCIDR9;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR9;
```

B.10.4 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 B-119: AArch64_trcidr2 bit assignments

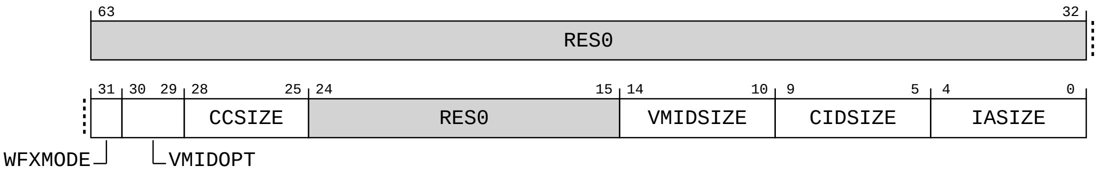


Table B-288: TRCIDR2 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31]	WFXMODE	Indicates whether WFI and WFE instructions are classified as P0 instructions: **0** WFI and WFE instructions are not classified as P0 instructions.	
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. **10** Virtual context identifier selection not supported. AArch64-TRCCONFIGR.VMIDOPT is RES1.	
[28:25]	CCSIZE	Indicates the size of the cycle counter. **0000** The cycle counter is 12 bits in length.	
[24:15]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. **`00100`** 32-bit Virtual context identifier size.	
[9:5]	CIDSIZE	Indicates the Context identifier size. **`00100`** 32-bit Context identifier size.	
[4:0]	IASIZE	Virtual instruction address size. **`01000`** 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 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;
```

B.10.5 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 B-120: AArch64_trcidr3 bit assignments

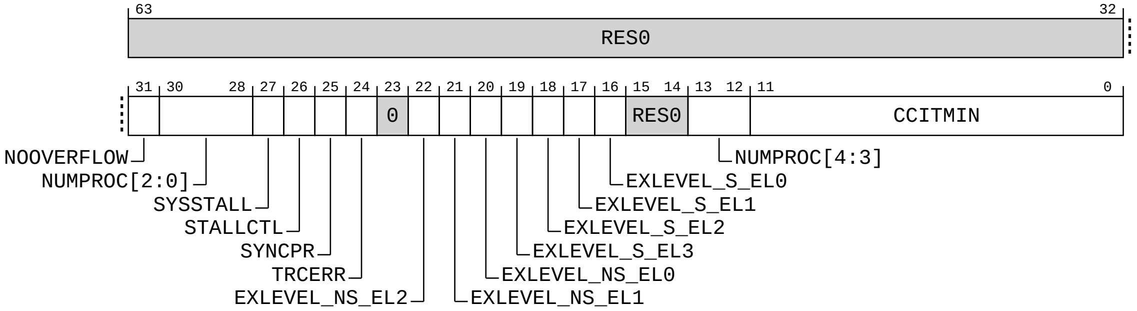


Table B-290: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. **0** Overflow prevention is not implemented.	
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing. **00000** The trace unit can trace one PE.	
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **1** Stalling of the PE is permitted.	
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **1** Stalling of the PE is implemented.	
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0** AArch64-TRCSYNCPR is read-write so software can change the synchronization period.	

Bits	Name	Description	Reset
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. **1** Forced tracing of System Error exceptions is implemented.	
[23]	RES0	Reserved	`0b0`
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 implemented. **1** Non-secure EL2 is implemented.	
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 implemented. **1** Non-secure EL1 is implemented.	
[20]	EXLEVEL_NS_EL0	Indicates if Non-secure EL0 implemented. **1** Non-secure EL0 is implemented.	
[19]	EXLEVEL_S_EL3	Indicates if Secure EL3 implemented. **1** Secure EL3 is implemented.	
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 implemented. **0** Secure EL2 is not implemented.	
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 implemented. **1** Secure EL1 is implemented.	
[16]	EXLEVEL_S_EL0	Indicates if Secure EL0 implemented. **1** Secure EL0 is implemented.	
[15:14]	RES0	Reserved	`0b0`
[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. **000000000100**	

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 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;
```

B.10.6 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 B-121: AArch64_trcidr4 bit assignments

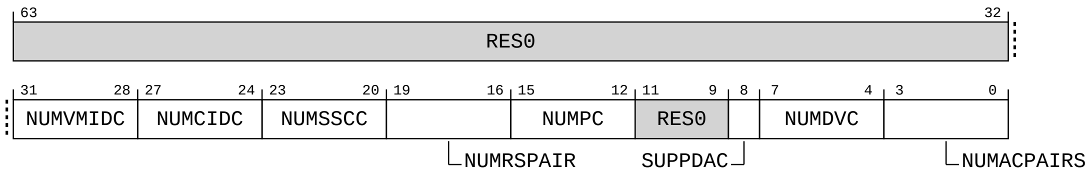


Table B-292: TRCIDR4 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. **0001** The implementation has one Virtual Context Identifier Comparator.	
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. **0001** The implementation has one Context Identifier Comparator.	
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. **0001** The implementation has one Single-shot Comparator Control.	
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **0111** The implementation has eight resource selector pairs.	
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **0000** No PE Comparator Inputs are available.	
[11:9]	RES0	Reserved	`0b0`
[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. **0** 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. **0000** No data value comparators implemented.	
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. **0100** 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 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;
```

B.10.7 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 B-122: AArch64_trcidr5 bit assignments

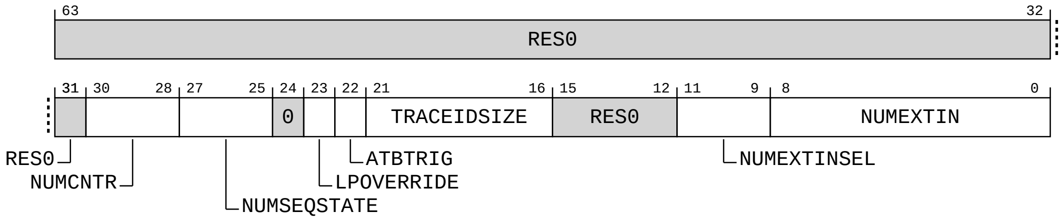


Table B-294: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	`0b0`
[30:28]	NUMCNTR	Indicates the number of Counters that are available for tracing. **`010`** Two Counters implemented.	

Bits	Name	Description	Reset
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. **`100`** Four Sequencer states are implemented.	
[24]	RES0	Reserved	`0b0`
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. **`1`** The trace unit supports Low-power Override Mode.	
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. **`1`** The implementation supports ATB triggers.	
[21:16]	TRACEIDSIZE	Indicates the trace ID width. **`000111`** 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. **`100`** 4 External Input Selector resources are available.	
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. **`111111111`** 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 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;
```

B.10.8 TRCIDR6, ID Register 6

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 B-123: AArch64_trcidr6 bit assignments

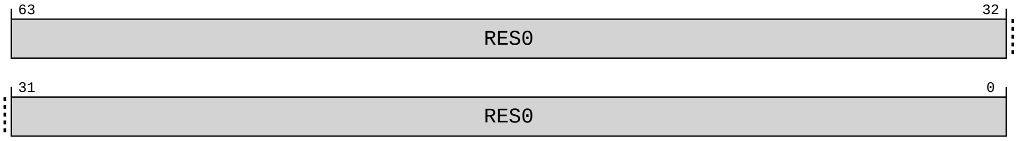


Table B-296: TRCIDR6 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	`0b0`

Access

MRS <Xt>, TRCIDR6

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR6	`0b10`	`0b001`	`0b0000`	`0b1110`	`0b111`

Accessibility

MRS <Xt>, TRCIDR6

```
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 CPTR_EL3.TTA == 1 then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return TRCIDR6;
    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 TRCIDR6;
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.TTA == 1 then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            return TRCIDR6;
```

B.10.9 TRCIDR7, ID Register 7

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 B-124: AArch64_trcidr7 bit assignments

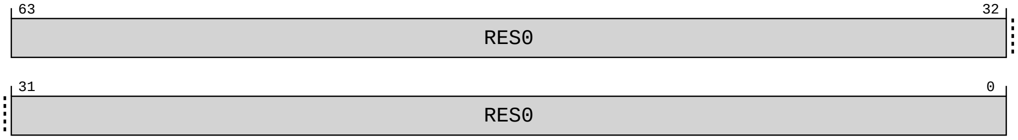


Table B-298: TRCIDR7 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

Access

MRS <Xt>, TRCIDR7

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR7	0b10	0b001	0b0000	0b1111	0b111

Accessibility

MRS <Xt>, TRCIDR7

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR7;
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 TRCIDR7;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR7;
```

B.10.10 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 B-125: AArch64_trcidr10 bit assignments

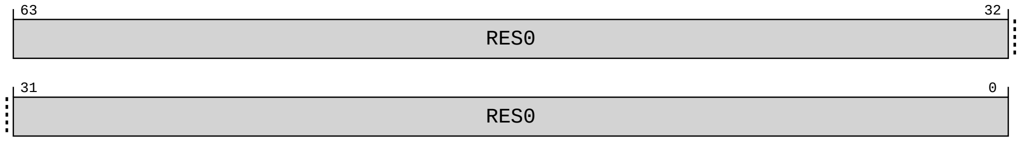


Table B-300: TRCIDR10 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

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;
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;
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 TRCIDR10;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR10;
```

B.10.11 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 B-126: AArch64_trcidr11 bit assignments

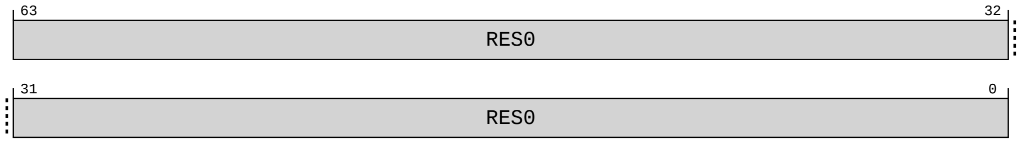


Table B-302: TRCIDR11 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	`0b0`

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 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;
```

B.10.12 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 B-127: AArch64_trcidr12 bit assignments

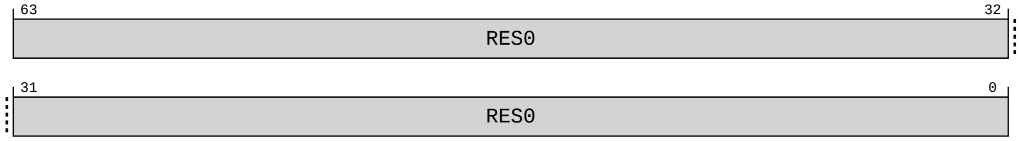


Table B-304: TRCIDR12 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

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 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;
```

B.10.13 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 B-128: AArch64_trcidr13 bit assignments

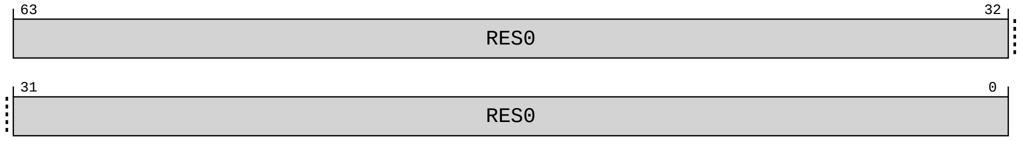


Table B-306: TRCIDR13 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

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;
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;
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 TRCIDR13;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR13;
```

B.10.14 TRCAUXCTLR, Auxiliary Control Register

The function of this register is **IMPLEMENTATION DEFINED**.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Trace

Reset value

`0x0`

Bit descriptions

Figure B-129: AArch64_trcauxctlr bit assignments

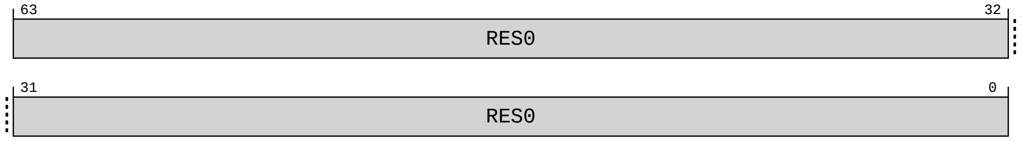


Table B-308: TRCAUXCTLR bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

Access

MRS <Xt>, TRCAUXCTLR

<systemreg>MRS <Xt>, TRCAUXCTLR	op0	op1	CRn	CRm	op2
TRCAUXCTLR	0b10	0b001	0b0000	0b0110	0b000

MRS MSR TRCAUXCTLR, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
TRCAUXCTLR	0b10	0b001	0b0000	0b0110	0b000

Accessibility

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCAUXCTLR;
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 TRCAUXCTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCAUXCTLR;
```

MSR TRCAUXCTLR, <Xt>

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCAUXCTLR = X[t];
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
        TRCAUXCTLR = X[t];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCAUXCTLR = X[t];
```

B.10.15 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 B-130: AArch64_trcidr0 bit assignments

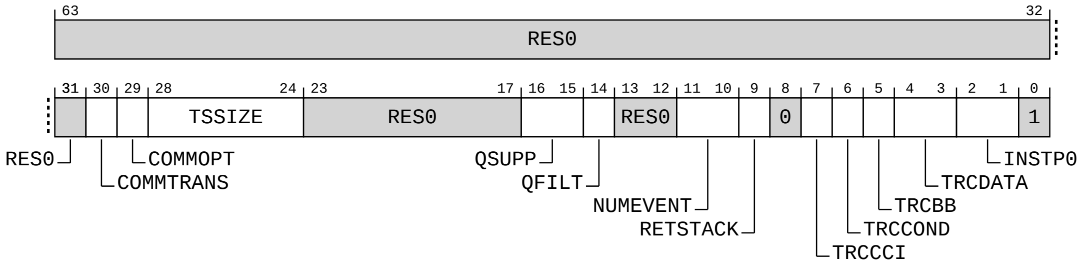


Table B-311: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	0b0
[30]	COMMTRANS	Transaction Start element behavior. **0** Transaction Start elements are P0 elements.	
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. **1** Commit mode 1.	
[28:24]	TSSIZE	Indicates that the trace unit implements Global timestamping and the size of the timestamp value. **01000** Global timestamping implemented with a 64-bit timestamp value.	
[23:17]	RES0	Reserved	0b0
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **00** Q element support is not implemented.	
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0** Q element filtering is not implemented.	
[13:12]	RES0	Reserved	0b0
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **11** The trace unit supports 4 ETEEvents.	
[9]	RETSTACK	Indicates if the trace unit supports the return stack. **1** Return stack implemented.	
[8]	RES0	Reserved	0b0
[7]	TRCCCI	Indicates if the trace unit implements cycle counting. **1** Cycle counting implemented.	

Bits	Name	Description	Reset
[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. `0` Conditional instruction tracing not implemented.	
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting. `1` 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. `00` Tracing of data addresses and data values is not implemented.	
[2:1]	INSTP0	Indicates if load and store instructions are P0 instructions. Load and store instructions as P0 instructions is not implemented in ETE and this field is reserved for other trace architectures. `00` Load and store instructions are not P0 instructions.	
[0]	RES1	Reserved	`0b1`

Access

MRS <Xt>, TRCIDR0

<systemreg>	op0	op1	CRn	CRm	op2
TRCIDR0	`0b10`	`0b001`	`0b0000`	`0b1000`	`0b111`

Accessibility

MRS <Xt>, TRCIDR0

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR0;
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 TRCIDR0;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR0;
```

B.10.16 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 B-131: AArch64_trcidr1 bit assignments

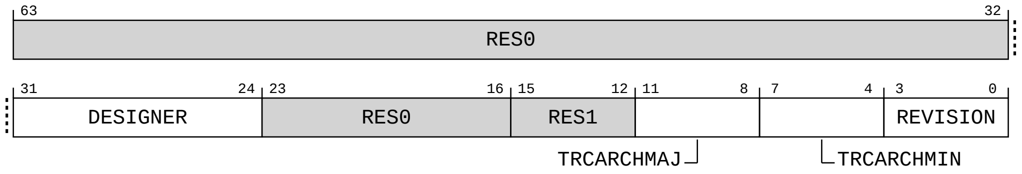


Table B-313: TRCIDR1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0b0`
[31:24]	DESIGNER	Indicates which company designed the trace unit. The permitted values of this field are the same as AArch64-MIDR_EL1.Implementer. **`01000001`** Arm Limited	
[23:16]	RES0	Reserved	`0b0`
[15:12]	RES1	Reserved	`0b1`
[11:8]	TRCARCHMAJ	Major architecture version. **`1111`** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to AArch64-TRCDEVARCH.	
[7:4]	TRCARCHMIN	Minor architecture version. **`1111`** 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. **`0000`** Revision 0	

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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;
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;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCIDR1;
```

B.10.17 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

64

Functional group

Trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-132: AArch64_trcdevarch bit assignments

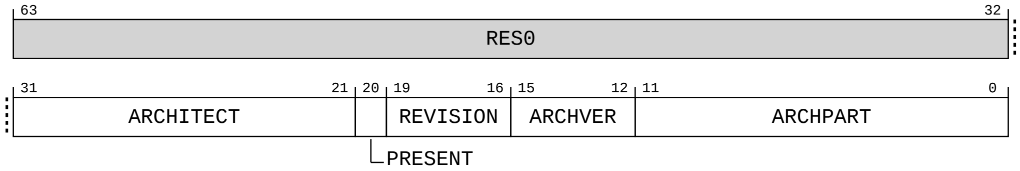


Table B-315: TRCDEVARCH bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:21]	ARCHITECT	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. **01000111011** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited. Other values are defined by the JEDEC JEP106 standard. This field reads as 0x23B.	
[20]	PRESENT	DEVARCH Present. Defines that the DEVARCH register is present. **1** Device Architecture information present.	
[19:16]	REVISION	Revision. Defines the architecture revision of the component. **0000** ETE Version 1.0. All other values are reserved.	
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **0101** ETE Version 1. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHVER is ARCHID[15:12]. This field reads as 0x5.	
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **101000010011** Arm PE trace architecture. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHPART is ARCHID[11:0]. This field reads as 0xA13.	

Access

MRS <Xt>, TRCDEVARCH

<systemreg>	op0	op1	CRn	CRm	op2
TRCDEVARCH	0b10	0b001	0b0111	0b1111	0b110

Accessibility

MRS <Xt>, TRCDEVARCH

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCDEVARCH;
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 TRCDEVARCH;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCDEVARCH;
```

B.10.18 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

64

Functional group

Trace

Reset value

0x0

Bit descriptions

Figure B-133: AArch64_trcdevid bit assignments

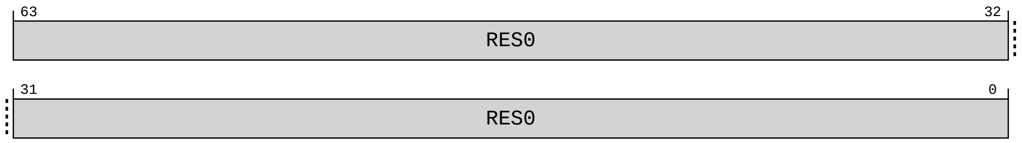


Table B-317: TRCDEVID bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

Access

MRS <Xt>, TRCDEVID

<systemreg>	op0	op1	CRn	CRm	op2
TRCDEVID	0b10	0b001	0b0111	0b0010	0b111

Accessibility

MRS <Xt>, TRCDEVID

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCDEVID;
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 TRCDEVID;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCDEVID;
```

B.10.19 TRCCLAIMSET, Claim Tag Set Register

In conjunction with AArch64-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

64

Functional group

Trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-134: AArch64_trcclaimset bit assignments

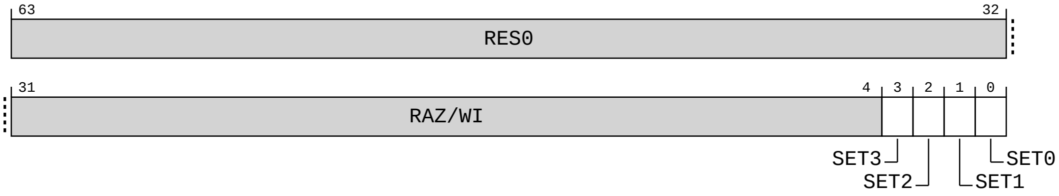


Table B-319: TRCCLAIMSET bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0b0`
[31:4]	RAZ/WI	Reserved	

[3]	SET3	Claim Tag Set. Indicates whether Claim Tag bit m is implemented, and is used to set Claim Tag bit m to 0b1.
		`0`
		On a read: Claim Tag bit m is not implemented.
		On a write: Ignored.
		`1`
		On a read: Claim Tag bit m is implemented.
		On a write: Set Claim Tag bit m to 0b1.	

[2]	SET2	Claim Tag Set. Indicates whether Claim Tag bit m is implemented, and is used to set Claim Tag bit m to 0b1.
		`0`
		On a read: Claim Tag bit m is not implemented.
		On a write: Ignored.
		`1`
		On a read: Claim Tag bit m is implemented.
		On a write: Set Claim Tag bit m to 0b1.	

Bits	Name	Description	Reset
[1]	SET1	Claim Tag Set. Indicates whether Claim Tag bit m is implemented, and is used to set Claim Tag bit m to 0b1. **0** On a read: Claim Tag bit m is not implemented. On a write: Ignored. **1** On a read: Claim Tag bit m is implemented. On a write: Set Claim Tag bit m to 0b1.	
[0]	SET0	Claim Tag Set. Indicates whether Claim Tag bit m is implemented, and is used to set Claim Tag bit m to 0b1. **0** On a read: Claim Tag bit m is not implemented. On a write: Ignored. **1** On a read: Claim Tag bit m is implemented. On a write: Set Claim Tag bit m to 0b1.	

Access

MRS <Xt>, TRCCLAIMSET

<systemreg>	op0	op1	CRn	CRm	op2
TRCCLAIMSET	0b10	0b001	0b0111	0b1000	0b110

MSR TRCCLAIMSET, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
TRCCLAIMSET	0b10	0b001	0b0111	0b1000	0b110

Accessibility

MRS <Xt>, TRCCLAIMSET

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCCLAIMSET;
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 TRCCLAIMSET;
elsif PSTATE.EL == EL3 then
```

```
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCCLAIMSET;
```

MSR TRCCLAIMSET, <Xt>

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCLAIMSET = X[t];
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
        TRCCLAIMSET = X[t];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCLAIMSET = X[t];
```

B.10.20 TRCCLAIMCLR, Claim Tag Clear Register

In conjunction with AArch64-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

64

Functional group

Trace

Reset value

See individual bit resets.

Bit descriptions

Figure B-135: AArch64_trcclaimclr bit assignments

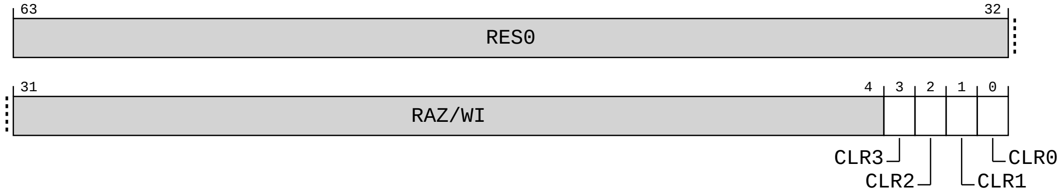


Table B-322: TRCCLAIMCLR bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0b0`
[31:4]	RAZ/WI	Reserved	
[3]	CLR3	Claim Tag Clear. Indicates the current status of the Claim Tag bit m, and is used to clear Claim Tag bit m to 0b0. **0** On a read: Claim Tag bit m is not set. On a write: Ignored. **1** On a read: Claim Tag bit m is set. On a write: Clear Claim tag bit m to 0b0.	
[2]	CLR2	Claim Tag Clear. Indicates the current status of the Claim Tag bit m, and is used to clear Claim Tag bit m to 0b0. **0** On a read: Claim Tag bit m is not set. On a write: Ignored. **1** On a read: Claim Tag bit m is set. On a write: Clear Claim tag bit m to 0b0.	
[1]	CLR1	Claim Tag Clear. Indicates the current status of the Claim Tag bit m, and is used to clear Claim Tag bit m to 0b0. **0** On a read: Claim Tag bit m is not set. On a write: Ignored. **1** On a read: Claim Tag bit m is set. On a write: Clear Claim tag bit m to 0b0.	

Bits	Name	Description	Reset
[0]	CLR0	Claim Tag Clear. Indicates the current status of the Claim Tag bit m, and is used to clear Claim Tag bit m to 0b0. **0** On a read: Claim Tag bit m is not set. On a write: Ignored. **1** On a read: Claim Tag bit m is set. On a write: Clear Claim tag bit m to 0b0.	

Access

MRS <Xt>, TRCCLAIMCLR

<systemreg>	op0	op1	CRn	CRm	op2
TRCCLAIMCLR	0b10	0b001	0b0111	0b1001	0b110

MSR TRCCLAIMCLR, <Xt>

<systemreg>	op0	op1	CRn	CRm	op2
TRCCLAIMCLR	0b10	0b001	0b0111	0b1001	0b110

Accessibility

MRS <Xt>, TRCCLAIMCLR

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCCLAIMCLR;
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 TRCCLAIMCLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        return TRCCLAIMCLR;
```

MSR TRCCLAIMCLR, <Xt>

```
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 CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCLAIMCLR = X[t];
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
        TRCCLAIMCLR = X[t];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == 1 then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCCLAIMCLR = X[t];
```

B.11 TRBE register summary

The summary table provides an overview of *TRace Buffer Extension* (TRBE) registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table B-325: TRBE register summary

Name	Op0	CRn	Op1	CRm	Op2	Reset	Width	Description
TRBIDR_EL1	3	C9	0	C11	7	See individual bit resets	64-bit	Trace Buffer ID Register

B.11.1 TRBIDR_EL1, Trace Buffer ID Register

Describes constraints on using the Trace Buffer Extension to software, including whether the Trace Buffer Extension can be programmed at the current Exception level.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

unknown

Reset value

See individual bit resets.

Bit descriptions

Figure B-136: AArch64_trbidr_el1 bit assignments

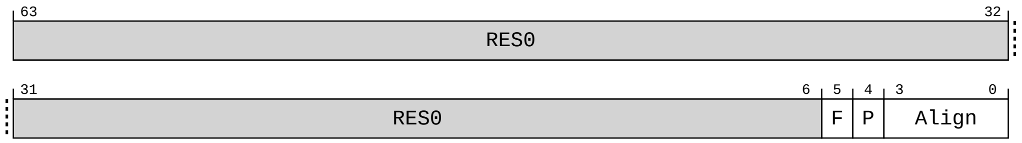


Table B-326: TRBIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[63:6]	RES0	Reserved	0b0
[5]	F	Flag Updates. Defines whether the address translation performed by the Trace Buffer Extension manages the Access Flag and dirty state. **1** Trace buffer address translation manages the Access Flag and dirty state in the same way as the MMU on this PE.	
[4]	P	Programming not allowed. The trace buffer is owned by a higher Exception level or by the other Security state. **0** The owning Exception level is the current Exception level or a lower Exception level, and the owning Security state is the current Security state. **1** The owning Exception level is a higher Exception level, or the owning Security state is not the current Security state. The value read from this field depends on the current Exception level and the values of AArch64-MDCR_EL3.NSTB and AArch64-MDCR_EL2.E2TB: • If EL3 is implemented and either AArch64-MDCR_EL3.NSTB == 0b00 or AArch64-MDCR_EL3.NSTB == 0b01, meaning the owning Security state is Secure state, this bit reads as one from: ◦ Non-secure EL2. ◦ Non-secure EL1. ◦ If Secure EL2 is implemented and enabled, and AArch64-MDCR_EL2.E2TB == 0b00, Secure EL1. • If EL3 is implemented and either AArch64-MDCR_EL3.NSTB == 0b10 or AArch64-MDCR_EL3.NSTB == 0b11, meaning the owning Security state is Non-secure state, this bit reads as one from: ◦ Secure EL1. ◦ If Secure EL2 is implemented, Secure EL2. ◦ If EL2 is implemented and AArch64-MDCR_EL2.E2TB == 0b00, Non-secure EL1. • Otherwise, this bit reads as zero.	
[3:0]	Align	Defines the minimum alignment constraint for writes to AArch64-TRBPTR_EL1 and AArch64-TRBTRG_EL1. **0110** 64 bytes.	

Access

MRS <Xt>, TRBIDR_EL1

<systemreg>	op0	op1	CRn	CRm	op2
TRBIDR_EL1	0b11	0b000	0b1001	0b1011	0b111

Accessibility

MRS <Xt>, TRBIDR_EL1

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    return TRBIDR_EL1;
elsif PSTATE.EL == EL2 then
    return TRBIDR_EL1;
elsif PSTATE.EL == EL3 then
    return TRBIDR_EL1;
```

Appendix C External registers

This appendix contains the descriptions for the Cortex®-A510 external registers.

C.1 MPMM register summary

The summary table provides an overview of memory-mapped *Maximum Power Mitigation Mechanism* (MPMM) registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table C-1: MPMM register summary

Offset	Name	Reset	Width	Description
0x000	CPUPPMCR	See individual bit resets.	64-bit	Global PPM Configuration Register
0x010	CPUMPMMCR	`0x0`	64-bit	Global MPMM Configuration Register

C.1.1 CPUPPMCR, Global PPM Configuration Register

This register controls global PPM features and allows discovery of some PPM implementation details.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

MPMM

Register offset

0x000

Reset value

See individual bit resets.

Bit descriptions

Figure C-1: ext_cpuppmcr bit assignments

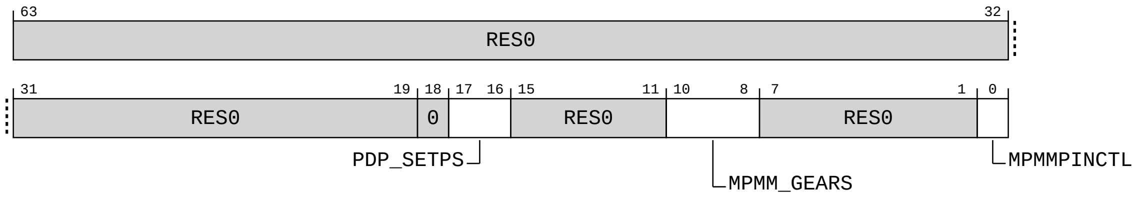


Table C-2: CPUPPMCR bit descriptions

Bits	Name	Description	Reset
[63:18]	RES0	Reserved	0b0
[17:16]	PDP_SETPS	Number of PDP Setpoints implemented **00** PDP is not implemented or enabled.	
[15:11]	RES0	Reserved	0b0
[10:8]	MPMM_GEARS	Number of MPMM Gears implemented **011** 3 MPMM are enabled.	
[7:1]	RES0	Reserved	0b0
[0]	MPMMPINCTL	MPMM Pin Control Enabled **0** MPMM control through SPR and utility bus. **1** MPMM control through pin only.	0b0

C.1.2 CPUMPMMCR, Global MPMM Configuration Register

This register is used to change MPMM gears or disable MPMM.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

MPMM

Register offset

0x010

Reset value

`0x0`

Bit descriptions

Figure C-2: ext_cpumpmmcr bit assignments

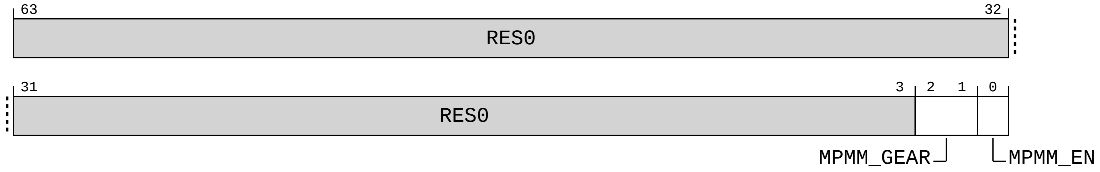


Table C-3: CPUMPMMCR bit descriptions

Bits	Name	Description	Reset
[63:3]	RES0	Reserved	`0b0`
[2:1]	MPMM_GEAR	MPMM Gear Select **00** Select MPMM Gear 0. **01** Select MPMM Gear 1. **10** Select MPMM Gear 2.	`0b00`
[0]	MPMM_EN	MPMM Master Enable **0** MPMM is disabled. **1** MPMM is enabled.	`0b0`

C.2 Memory-mapped PMU register summary

The summary table provides an overview of memory-mapped *Performance Monitoring Unit* (PMU) registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table C-4: 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
0x614	PMOVSSR	See individual bit resets	32-bit	PMU Overflow Status Snapshot Register
0x618	PMCCNTSR	See individual bit resets	64-bit	PMU Cycle Counter Snapshot Register

Offset	Name	Reset	Width	Description
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
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_EL0	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
0xFBC	PMDEVARCH	See individual bit resets	32-bit	Performance Monitors Device Architecture register
0xFC8	PMDEVID	See individual bit resets	32-bit	Performance Monitors Device ID register
0xFCC	PMDEVTYPE	See individual bit resets	32-bit	Performance Monitors Device Type register
0xFD0	PMPIDR4	See individual bit resets	32-bit	Performance Monitors Peripheral Identification Register 4
0xFE0	PMPIDR0	See individual bit resets	32-bit	Performance Monitors Peripheral Identification Register 0
0xFE4	PMPIDR1	See individual bit resets	32-bit	Performance Monitors Peripheral Identification Register 1
0xFE8	PMPIDR2	See individual bit resets	32-bit	Performance Monitors Peripheral Identification Register 2
0xFEC	PMPIDR3	See individual bit resets	32-bit	Performance Monitors Peripheral Identification Register 3
0xFF0	PMCIDR0	See individual bit resets	32-bit	Performance Monitors Component Identification Register 0
0xFF4	PMCIDR1	See individual bit resets	32-bit	Performance Monitors Component Identification Register 1
0xFF8	PMCIDR2	See individual bit resets	32-bit	Performance Monitors Component Identification Register 2
0xFFC	PMCIDR3	See individual bit resets	32-bit	Performance Monitors Component Identification Register 3

C.2.1 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 C-3: ext_pmpcssr bit assignments

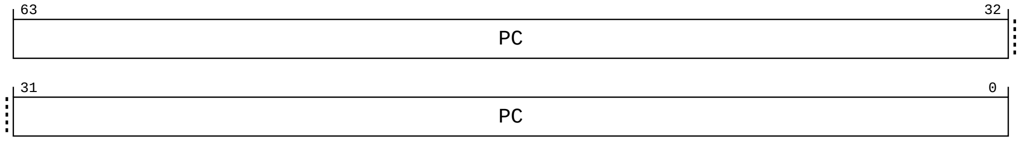


Table C-5: PMPCSSR bit descriptions

Bits	Name	Description	Reset
[63:0]	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. **Note:** The Arm architecture does not define recently executed.	

C.2.2 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 C-4: ext_pmcidssr bit assignments

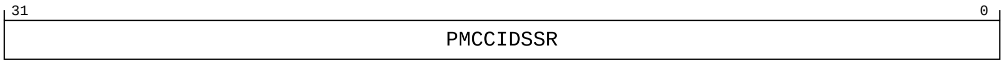


Table C-6: PMCIDSSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMCCIDSSR	PMCIDSR sample. Sampled CONTEXTIDR_EL1 snapshot.	

C.2.3 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 C-5: ext_pmcid2ssr bit assignments

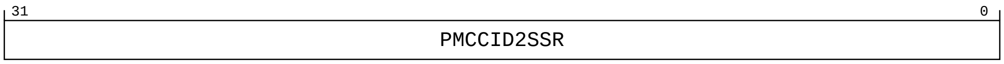


Table C-7: PMCID2SSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMCCID2SSR	PMCID2SR sample. Sampled CONTEXTIDR_EL2 snapshot.	

C.2.4 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 C-6: ext_pmsssr bit assignments

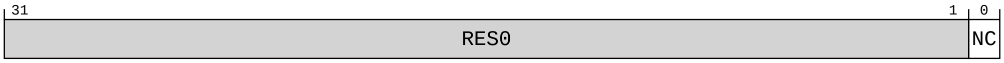


Table C-8: PMSSSR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	`0b0`

Bits	Name	Description	Reset
[0]	NC	No capture. Indicates whether the PMU counters have been captured. **0** PMU counters captured. **1** PMU counters not captured. The event counters are only not captured by the PE in the event of a security violation. The external Monitor is responsible for keeping track of whether it managed to capture the snapshot registers from the PE. PMSSR.NC does not reflect the status of the captured Program Counter Sample registers. PMSSR.NC is reset to 1 by PE Warm reset, but is overwritten at the first capture. Tools need to be aware that capturing over reset or power-down might lose data, as they are reliant on software saving and restoring the PMU state (including PMSSCR). There is no sampled sticky reset bit.	`0b1`

C.2.5 PMOVSSR, PMU Overflow Status Snapshot Register

Captured copy of PMOVSR. Once captured, the value in PMOVSSR is unaffected by writes to PMOVSSET_EL0 and PMOVSCLR_EL0.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

PMU

Register offset

0x614

Reset value

See individual bit resets.

Bit descriptions

Figure C-7: ext_pmovssr bit assignments

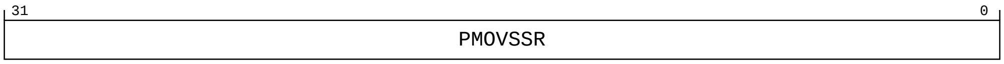


Table C-9: PMOVSSR bit descriptions

Bits	Name	Description	Reset
[31:0]	PMOVSSR	PMOVSR sample. Sampled overflow status.	

C.2.6 PMCCNTSR, PMU Cycle Counter Snapshot Register

Captured copy of PMCCNTR_EL0. Once captured, the value in PMCCNTSR is unaffected by writes to PMCCNTR_EL0 and PMCR_EL0.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 C-8: ext_pmccntsr bit assignments

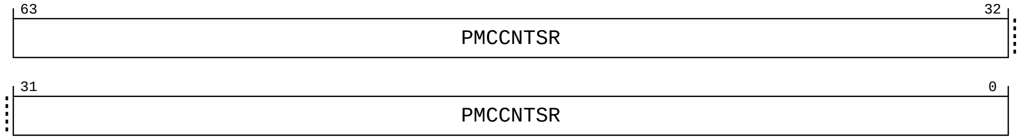


Table C-10: PMCCNTSR bit descriptions

Bits	Name	Description	Reset
[63:0]	PMCCNTSR	PMCCNTR_EL0 sample. Sampled cycle count.	

C.2.7 PMEVCNTSR0, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.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 C-9: ext_pmevcntsr0 bit assignments

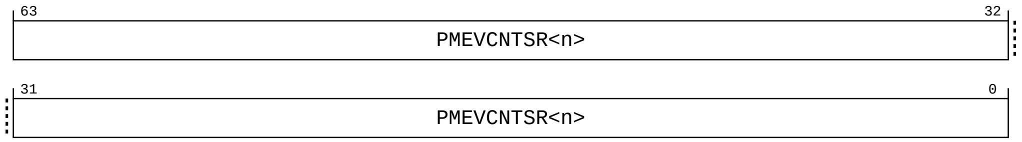


Table C-11: PMEVCNTSR0 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	

C.2.8 PMEVCNTSR1, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.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 C-10: ext_pmevcntsr1 bit assignments

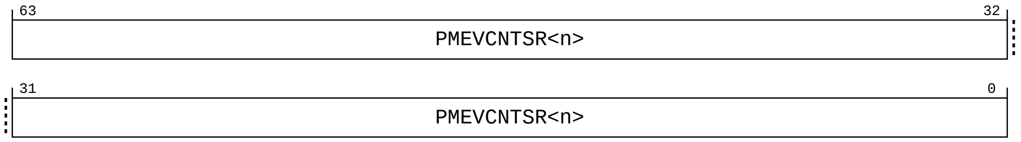


Table C-12: PMEVCNTSR1 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	

C.2.9 PMEVCNTSR2, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.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 C-11: ext_pmevcntsr2 bit assignments

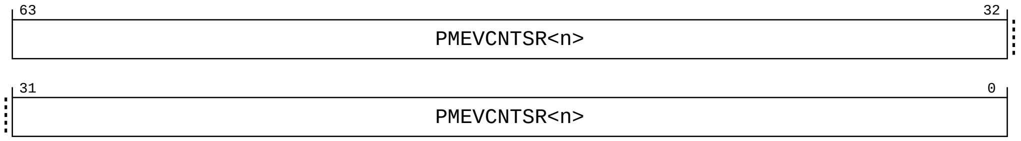


Table C-13: PMEVCNTSR2 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	

C.2.10 PMEVCNTSR3, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.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 C-12: ext_pmevcntsr3 bit assignments

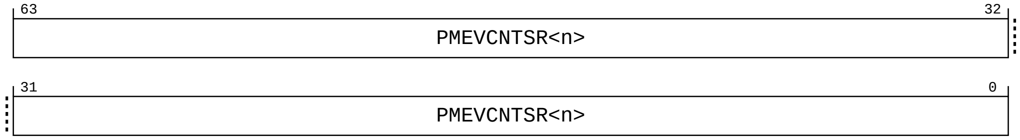


Table C-14: PMEVCNTSR3 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	

C.2.11 PMEVCNTSR4, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.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 C-13: ext_pmevcntsr4 bit assignments

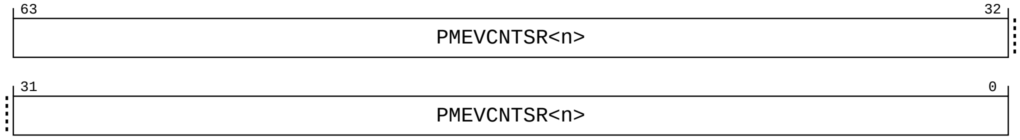


Table C-15: PMEVCNTSR4 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	

C.2.12 PMEVCNTSR5, PMU Event Counter Snapshot Register

Captured copy of PMEVCNTR<n>_EL0. Once captured, the value in PMSSEVCNTR<n> is unaffected by writes to PMSSEVCNTR<n>_EL0 and PMCR_EL0.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 C-14: ext_pmevcntsr5 bit assignments

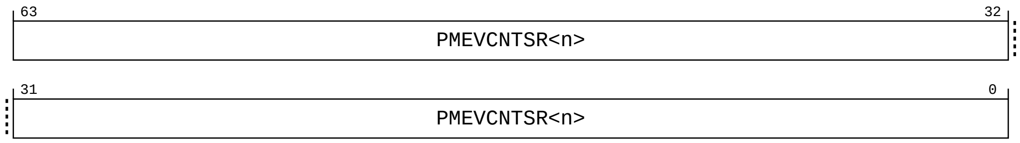


Table C-16: PMEVCNTSR5 bit descriptions

Bits	Name	Description	Reset
[63:0]	PMEVCNTSR<n>	PMEVCNTR<n>_EL0 sample. Sampled event count.	

C.2.13 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 C-15: ext_pmsscr bit assignments

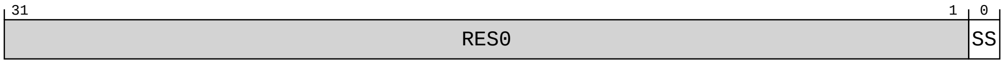


Table C-17: PMSSCR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	0b0

Bits	Name	Description	Reset

[0]	SS	Capture now.
		`0`
		Ignored.
		`1`
		Initiate a capture immediately.	

C.2.14 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 C-16: ext_pmcfgr bit assignments

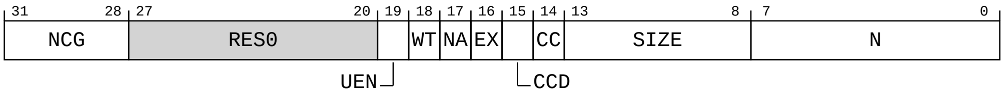


Table C-18: PMCFGR bit descriptions

Bits	Name	Description	Reset
[31:28]	NCG	This feature is not supported, so this field is RAZ.	
[27:20]	RES0	Reserved	`0b0`
[19]	UEN	User-mode Enable Register supported. AArch64-PMUSERENR_EL0 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.	

Bits	Name	Description	Reset
[16]	EX	Export supported. Value is **IMPLEMENTATION DEFINED**. **0** ext-PMCR_EL0.X is RES0.	
[15]	CCD	Cycle counter has prescale. This field is RAZ **0** ext-PMCR_EL0.D is RES0.	
[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_EL0. The maximum number of event counters is 31. **00000110** Six PMU Counters Implemented	

C.2.15 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

32

Functional group

PMU

Register offset

0xE04

Reset value

See individual bit resets.

Bit descriptions

Figure C-17: ext_pmcr_el0 bit assignments

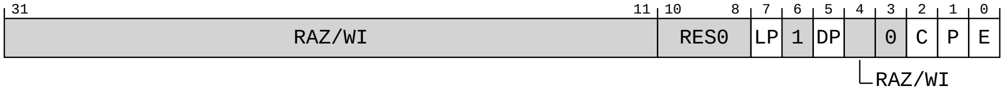


Table C-19: PMCR_EL0 bit descriptions

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. **0** Event counter overflow on increment that causes unsigned overflow of ext-PMEVCNTR<n>_EL0[31:0]. **1** Event counter overflow on increment that causes unsigned overflow of ext-PMEVCNTR<n>_EL0[63:0]. If EL2 is implemented and AArch64-MDCR_EL2.HPMN is less than PMCR_EL0.N, this bit does not affect the operation of event counters in the range [AArch64-MDCR_EL2.HPMN:(PMCR_EL0.N-1)]. **Note:** 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.	
[6]	RES1	Reserved	`0b1`
[5]	DP	Disable cycle counter when event counting is prohibited. The possible values of this bit are: **0** Cycle counting by ext-PMCCNTR_EL0 is not affected by this bit. **1** When event counting for counters in the range [0..(AArch64-MDCR_EL2.HPMN-1)] is prohibited, cycle counting by ext-PMCCNTR_EL0 is disabled. For more information see 'Prohibiting event counting'.	
[4]	RAZ/WI	Reserved	
[3]	RES0	Reserved	`0b0`

Bits	Name	Description	Reset
[2]	C	Cycle counter reset. The effects of writing to this bit are: **0** No action. **1** Reset ext-PMCCNTR_EL0 to zero. This bit is always RAZ. **Note:** Resetting ext-PMCCNTR_EL0 does not change the cycle counter overflow bit.	
[1]	P	Event counter reset. The effects of writing to this bit are: **0** No action. **1** Reset all event counters, not including ext-PMCCNTR_EL0, to zero. This bit is always RAZ. **Note:** Resetting the event counters does not change the event counter overflow bits. If ARMv8.5-PMU is implemented, the value of AArch64-MDCR_EL2.HLP, or PMCR_EL0.LP is ignored and bits [63:0] of all event counters are reset.	
[0]	E	Enable. **0** All event counters in the range [0..(PMN-1)] and ext-PMCCNTR_EL0, are disabled. **1** All event counters in the range [0..(PMN-1)] and ext-PMCCNTR_EL0, are enabled by ext-PMCNTENSET_EL0. If EL2 is implemented then: If EL2 is using AArch64, PMN is AArch64-MDCR_EL2.HPMN. If PMN is less than PMCR_EL0.N, this bit does not affect the operation of event counters in the range [PMN..(PMCR_EL0.N-1)]. If EL2 is not implemented, PMN is PMCR_EL0.N. **Note:** The effect of the following fields on the operation of this bit applies if EL2 is implemented regardless of whether EL2 is enabled in the current Security state: • AArch64-MDCR_EL2.HPMN. See the description of AArch64-MDCR_EL2.HPMN for more information.	

C.2.16 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_EL0.

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 C-18: ext_pmceid0 bit assignments

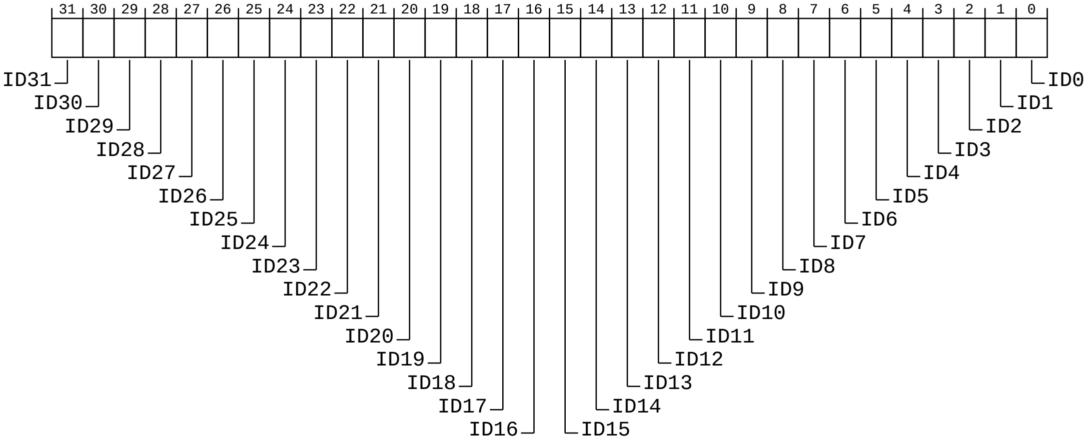


Table C-20: PMCEID0 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x1f) L1D_CACHE_ALLOCATE **0** The common event is not implemented, or not counted.	
[30]	ID30	ID30 corresponds to common event (0x1e) CHAIN **1** The common event is implemented.	
[29]	ID29	ID29 corresponds to common event (0x1d) BUS_CYCLES **1** The common event is implemented.	
[28]	ID28	ID28 corresponds to common event (0x1c) TTBR_WRITE_RETIRED **1** The common event is implemented.	
[27]	ID27	ID27 corresponds to common event (0x1b) INST_SPEC **1** The common event is implemented.	
[26]	ID26	ID26 corresponds to common event (0x1a) MEMORY_ERROR **1** The common event is implemented.	
[25]	ID25	ID25 corresponds to common event (0x19) BUS_ACCESS **1** The common event is implemented.	

Bits	Name	Description	Reset
[24]	ID24	ID24 corresponds to common event (0x18) L2D_CACHE_WB **0** The common event is not implemented, or not counted. This value is reported if the Cortex®-A510 complex is configured without an L2 cache. **1** The common event is implemented. This value is reported if the Cortex®-A510 complex is configured with an L2 cache.	
[23]	ID23	ID23 corresponds to common event (0x17) L2D_CACHE_REFILL **0** The common event is not implemented, or not counted. This value is reported if the Cortex®-A510 complex is configured without an L2 cache. **1** The common event is implemented. This value is reported if the Cortex®-A510 complex is configured with an L2 cache.	
[22]	ID22	ID22 corresponds to common event (0x16) L2D_CACHE **0** The common event is not implemented, or not counted. This value is reported if both the Cortex®-A510 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **1** The common event is implemented. This value is reported if either the Cortex®-A510 complex is configured with an L2 cache or the DSU is configured with an L3 cache.	
[21]	ID21	ID21 corresponds to common event (0x15) L1D_CACHE_WB **1** The common event is implemented.	
[20]	ID20	ID20 corresponds to common event (0x14) L1I_CACHE **1** The common event is implemented.	
[19]	ID19	ID19 corresponds to common event (0x13) MEM_ACCESS **1** The common event is implemented.	
[18]	ID18	ID18 corresponds to common event (0x12) BR_PRED **1** The common event is implemented.	
[17]	ID17	ID17 corresponds to common event (0x11) CPU_CYCLES **1** The common event is implemented.	
[16]	ID16	ID16 corresponds to common event (0x10) BR_MIS_PRED **1** The common event is implemented.	
[15]	ID15	ID15 corresponds to common event (0xf) UNALIGNED_LDST_RETIRED **0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[14]	ID14	ID14 corresponds to common event (0xe) BR_RETURN_RETIRED **1** The common event is implemented.	
[13]	ID13	ID13 corresponds to common event (0xd) BR_IMMED_RETIRED **1** The common event is implemented.	
[12]	ID12	ID12 corresponds to common event (0xc) PC_WRITE_RETIRED **1** The common event is implemented.	
[11]	ID11	ID11 corresponds to common event (0xb) CID_WRITE_RETIRED **1** The common event is implemented.	
[10]	ID10	ID10 corresponds to common event (0xa) EXC_RETURN **1** The common event is implemented.	
[9]	ID9	ID9 corresponds to common event (0x9) EXC_TAKEN **1** The common event is implemented.	
[8]	ID8	ID8 corresponds to common event (0x8) INST_RETIRED **1** The common event is implemented.	
[7]	ID7	ID7 corresponds to common event (0x7) ST_RETIRED **1** The common event is implemented.	
[6]	ID6	ID6 corresponds to common event (0x6) LD_RETIRED **1** The common event is implemented.	
[5]	ID5	ID5 corresponds to common event (0x5) L1D_TLB_REFILL **1** The common event is implemented.	
[4]	ID4	ID4 corresponds to common event (0x4) L1D_CACHE **1** The common event is implemented.	
[3]	ID3	ID3 corresponds to common event (0x3) L1D_CACHE_REFILL **1** The common event is implemented.	
[2]	ID2	ID2 corresponds to common event (0x2) L1I_TLB_REFILL **1** The common event is implemented.	
[1]	ID1	ID1 corresponds to common event (0x1) L1I_CACHE_REFILL **1** The common event is implemented.	

Bits	Name	Description	Reset
[0]	ID0	ID0 corresponds to common event (0x0) SW_INCR **1** The common event is implemented.	

C.2.17 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_EL0.

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 C-19: ext_pmceid1 bit assignments

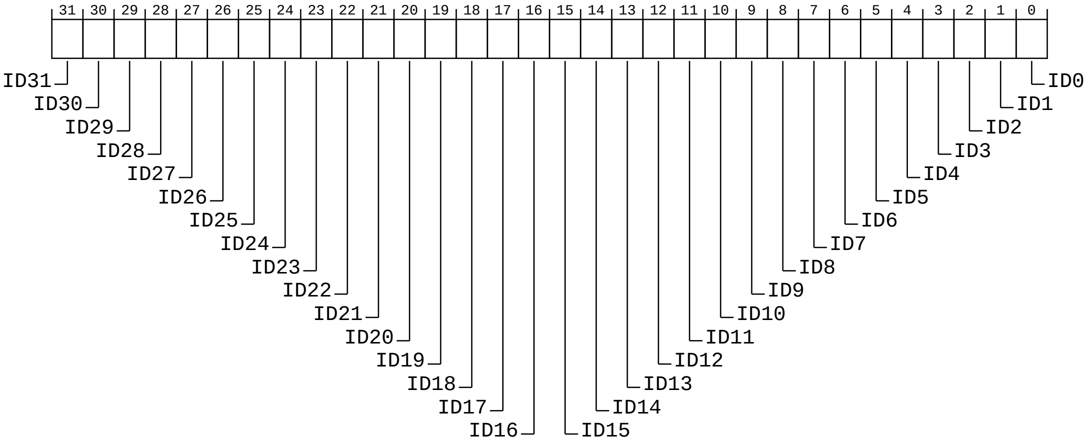


Table C-21: PMCEID1 bit descriptions

Bits	Name	Description	Reset
[31]	ID31	ID31 corresponds to common event (0x3f) STALL_SLOT **1** The common event is implemented.	
[30]	ID30	ID30 corresponds to common event (0x3e) STALL_SLOT_FRONTEND **1** The common event is implemented.	
[29]	ID29	ID29 corresponds to common event (0x3d) STALL_SLOT_BACKEND **1** The common event is implemented.	
[28]	ID28	ID28 corresponds to common event (0x3c) STALL **1** The common event is implemented.	
[27]	ID27	ID27 corresponds to common event (0x3b) OP_SPEC **1** The common event is implemented.	
[26]	ID26	ID26 corresponds to common event (0x3a) OP_RETIRED **1** The common event is implemented.	
[25]	ID25	ID25 corresponds to common event (0x39) L1D_CACHE_LMISS_RD **1** The common event is implemented.	
[24]	ID24	ID24 corresponds to common event (0x38) REMOTE_ACCESS_RD **1** The common event is implemented.	

Bits	Name	Description	Reset
[23]	ID23	ID23 corresponds to common event (0x37) LL_CACHE_MISS_RD **1** The common event is implemented.	
[22]	ID22	ID22 corresponds to common event (0x36) LL_CACHE_RD **1** The common event is implemented.	
[21]	ID21	ID21 corresponds to common event (0x35) ITLB_WLK **1** The common event is implemented.	
[20]	ID20	ID20 corresponds to common event (0x34) DTLB_WLK **1** The common event is implemented.	
[19]	ID19	ID19 corresponds to a Reserved Event event (0x33) **0** The common event is not implemented, or not counted.	
[18]	ID18	ID18 corresponds to a Reserved Event event (0x32) **0** The common event is not implemented, or not counted.	
[17]	ID17	ID17 corresponds to common event (0x31) REMOTE_ACCESS **0** The common event is not implemented, or not counted.	
[16]	ID16	ID16 corresponds to common event (0x30) L2I_TLB **0** The common event is not implemented, or not counted.	
[15]	ID15	ID15 corresponds to common event (0x2f) L2D_TLB **1** The common event is implemented.	
[14]	ID14	ID14 corresponds to common event (0x2e) L2I_TLB_REFILL **0** The common event is not implemented, or not counted.	
[13]	ID13	ID13 corresponds to common event (0x2d) L2D_TLB_REFILL **1** The common event is implemented.	
[12]	ID12	ID12 corresponds to common event (0x2c) Reserved **0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[11]	ID11	ID11 corresponds to common event (0x2b) L3D_CACHE **0** The common event is not implemented, or not counted. This value is reported if either the Cortex®-A510 complex is configured without an L2 cache or the DSU is configured without an L3 cache. **1** The common event is implemented. This value is reported if both the Cortex®-A510 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	
[10]	ID10	ID10 corresponds to common event (0x2a) L3D_CACHE_REFILL **0** The common event is not implemented, or not counted.	
[9]	ID9	ID9 corresponds to common event (0x29) L3D_CACHE_ALLOCATE **0** The common event is not implemented, or not counted.	
[8]	ID8	ID8 corresponds to common event (0x28) L2I_CACHE_REFILL **0** The common event is not implemented, or not counted.	
[7]	ID7	ID7 corresponds to common event (0x27) L2I_CACHE **0** The common event is not implemented, or not counted.	
[6]	ID6	ID6 corresponds to common event (0x26) L1I_TLB **1** The common event is implemented.	
[5]	ID5	ID5 corresponds to common event (0x25) L1D_TLB **1** The common event is implemented.	
[4]	ID4	ID4 corresponds to common event (0x24) STALL_BACKEND **1** The common event is implemented.	
[3]	ID3	ID3 corresponds to common event (0x23) STALL_FRONTEND **1** The common event is implemented.	
[2]	ID2	ID2 corresponds to common event (0x22) BR_MIS_PRED_RETIRED **1** The common event is implemented.	
[1]	ID1	ID1 corresponds to common event (0x21) BR_RETIRED **1** The common event is implemented.	

Bits	Name	Description	Reset
[0]	ID0	ID0 corresponds to common event (0x20) L2D_CACHE_ALLOCATE **0** The common event is not implemented, or not counted. This value is reported if the Cortex®-A510 complex is configured without an L2 cache. **1** The common event is implemented. This value is reported if the Cortex®-A510 complex is configured with an L2 cache.	

C.2.18 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.

Note

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

For more information about the common events and the use of the PMCEIDn registers see 'The 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 C-20: ext_pmceid2 bit assignments

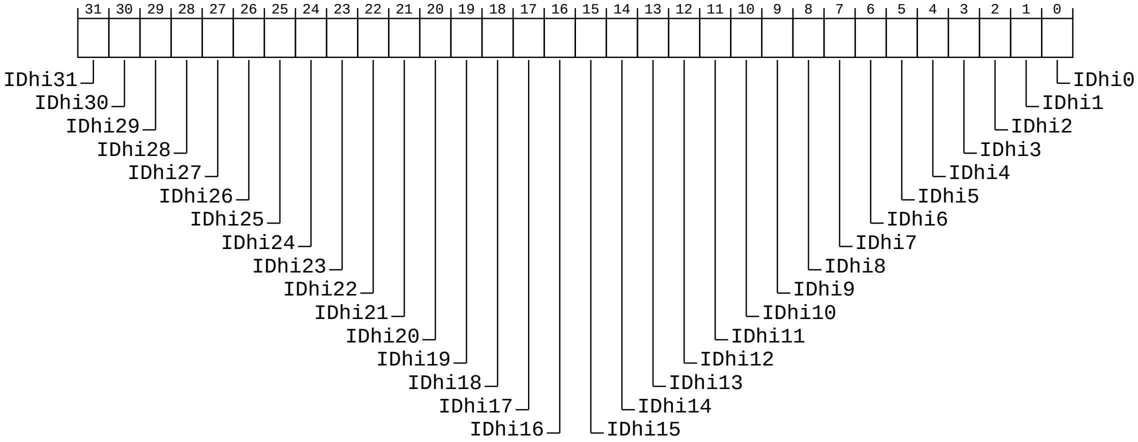


Table C-22: PMCEID2 bit descriptions

Bits	Name	Description	Reset
[31]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x401f) 0 The common event is not implemented, or not counted.	
[30]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x401e) 0 The common event is not implemented, or not counted.	
[29]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x401d) 0 The common event is not implemented, or not counted.	
[28]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x401c) 0 The common event is not implemented, or not counted.	
[27]	IDhi27	IDhi27 corresponds to common event (0x401b) CTI_TRIGOUT7 1 The common event is implemented.	
[26]	IDhi26	IDhi26 corresponds to common event (0x401a) CTI_TRIGOUT6 1 The common event is implemented.	
[25]	IDhi25	IDhi25 corresponds to common event (0x4019) CTI_TRIGOUT5 1 The common event is implemented.	
[24]	IDhi24	IDhi24 corresponds to common event (0x4018) CTI_TRIGOUT4 1 The common event is implemented.	

Bits	Name	Description	Reset
[23]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4017) `0` The common event is not implemented, or not counted.	
[22]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4016) `0` The common event is not implemented, or not counted.	
[21]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4015) `0` The common event is not implemented, or not counted.	
[20]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4014) `0` The common event is not implemented, or not counted.	
[19]	IDhi19	IDhi19 corresponds to common event (0x4013) TRCEXTOUT3 `1` The common event is implemented.	
[18]	IDhi18	IDhi18 corresponds to common event (0x4012) TRCEXTOUT2 `1` The common event is implemented.	
[17]	IDhi17	IDhi17 corresponds to common event (0x4011) TRCEXTOUT1 `1` The common event is implemented.	
[16]	IDhi16	IDhi16 corresponds to common event (0x4010) TRCEXTOUT0 `1` The common event is implemented.	
[15]	IDhi15	IDhi15 corresponds to common event (0x400f) PMU_HOVFS `0` The common event is not implemented, or not counted.	
[14]	IDhi14	IDhi14 corresponds to common event (0x400e) TRB_TRIG `1` The common event is implemented.	
[13]	IDhi13	IDhi13 corresponds to common event (0x400d) PMU_OVFS `0` The common event is not implemented, or not counted.	
[12]	IDhi12	IDhi12 corresponds to common event (0x400c) TRB_WRAP `1` The common event is implemented.	

Bits	Name	Description	Reset
[11]	IDhi11	IDhi11 corresponds to common event (0x400b) L3D_CACHE_LMISS_RD **0** The common event is not implemented, or not counted. This value is reported if either the Cortex®-A510 complex is configured without an L2 cache or the DSU is configured without an L3 cache. **1** The common event is implemented. This value is reported if both the Cortex®-A510 complex is configured with an L2 cache and the DSU is configured with an L3 cache.	
[10]	IDhi10	IDhi10 corresponds to common event (0x400a) L2I_CACHE_LMISS **0** The common event is not implemented, or not counted.	
[9]	IDhi9	IDhi9 corresponds to common event (0x4009) L2D_CACHE_LMISS_RD **0** The common event is not implemented, or not counted. This value is reported if both the Cortex®-A510 complex is configured without an L2 cache and the DSU is configured without an L3 cache. **1** The common event is implemented. This value is reported if either the Cortex®-A510 complex is configured with an L2 cache or the DSU is configured with an L3 cache.	
[8]	IDhi8	IDhi8 corresponds to common event (0x4008) Reserved **0** The common event is not implemented, or not counted.	
[7]	IDhi7	IDhi7 corresponds to common event (0x4007) Reserved **0** The common event is not implemented, or not counted.	
[6]	IDhi6	IDhi6 corresponds to common event (0x4006) L1I_CACHE_LMISS **1** The common event is implemented.	
[5]	IDhi5	IDhi5 corresponds to common event (0x4005) STALL_BACKEND_MEM **1** The common event is implemented.	
[4]	IDhi4	IDhi4 corresponds to common event (0x4004) CNT_CYCLES **0** The common event is not implemented, or not counted.	
[3]	IDhi3	IDhi3 corresponds to common event (0x4003) SAMPLE_COLLISION **0** The common event is not implemented, or not counted.	
[2]	IDhi2	IDhi2 corresponds to common event (0x4002) SAMPLE_FILTRATE **0** The common event is not implemented, or not counted.	
[1]	IDhi1	IDhi1 corresponds to common event (0x4001) SAMPLE_FEED **0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[0]	IDhi0	IDhi0 corresponds to common event (0x4000) SAMPLE_POP **0** The common event is not implemented, or not counted.	

C.2.19 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.

Note

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

For more information about the common events and the use of the PMCEIDn registers see 'The 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 C-21: ext_pmceid3 bit assignments

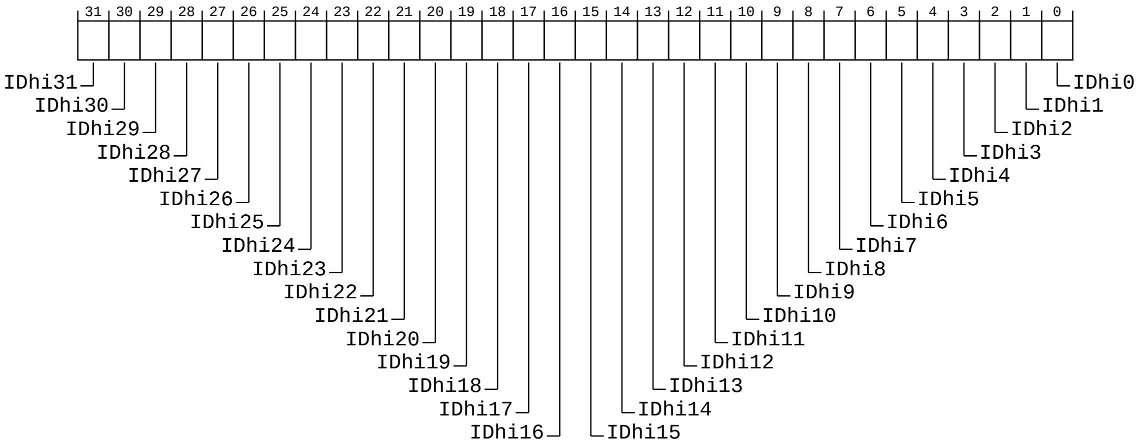


Table C-23: PMCEID3 bit descriptions

Bits	Name	Description	Reset
[31]	IDhi31	IDhi31 corresponds to a Reserved Event event (0x403f) 0 The common event is not implemented, or not counted.	
[30]	IDhi30	IDhi30 corresponds to a Reserved Event event (0x403e) 0 The common event is not implemented, or not counted.	
[29]	IDhi29	IDhi29 corresponds to a Reserved Event event (0x403d) 0 The common event is not implemented, or not counted.	
[28]	IDhi28	IDhi28 corresponds to a Reserved Event event (0x403c) 0 The common event is not implemented, or not counted.	
[27]	IDhi27	IDhi27 corresponds to a Reserved Event event (0x403b) 0 The common event is not implemented, or not counted.	
[26]	IDhi26	IDhi26 corresponds to a Reserved Event event (0x403a) 0 The common event is not implemented, or not counted.	
[25]	IDhi25	IDhi25 corresponds to a Reserved Event event (0x4039) 0 The common event is not implemented, or not counted.	
[24]	IDhi24	IDhi24 corresponds to a Reserved Event event (0x4038) 0 The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[23]	IDhi23	IDhi23 corresponds to a Reserved Event event (0x4037) **0** The common event is not implemented, or not counted.	
[22]	IDhi22	IDhi22 corresponds to a Reserved Event event (0x4036) **0** The common event is not implemented, or not counted.	
[21]	IDhi21	IDhi21 corresponds to a Reserved Event event (0x4035) **0** The common event is not implemented, or not counted.	
[20]	IDhi20	IDhi20 corresponds to a Reserved Event event (0x4034) **0** The common event is not implemented, or not counted.	
[19]	IDhi19	IDhi19 corresponds to a Reserved Event event (0x4033) **0** The common event is not implemented, or not counted.	
[18]	IDhi18	IDhi18 corresponds to a Reserved Event event (0x4032) **0** The common event is not implemented, or not counted.	
[17]	IDhi17	IDhi17 corresponds to a Reserved Event event (0x4031) **0** The common event is not implemented, or not counted.	
[16]	IDhi16	IDhi16 corresponds to a Reserved Event event (0x4030) **0** The common event is not implemented, or not counted.	
[15]	IDhi15	IDhi15 corresponds to a Reserved Event event (0x402f) **0** The common event is not implemented, or not counted.	
[14]	IDhi14	IDhi14 corresponds to a Reserved Event event (0x402e) **0** The common event is not implemented, or not counted.	
[13]	IDhi13	IDhi13 corresponds to a Reserved Event event (0x402d) **0** The common event is not implemented, or not counted.	
[12]	IDhi12	IDhi12 corresponds to a Reserved Event event (0x402c) **0** The common event is not implemented, or not counted.	
[11]	IDhi11	IDhi11 corresponds to a Reserved Event event (0x402b) **0** The common event is not implemented, or not counted.	
[10]	IDhi10	IDhi10 corresponds to a Reserved Event event (0x402a) **0** The common event is not implemented, or not counted.	

Bits	Name	Description	Reset
[9]	IDhi9	IDhi9 corresponds to a Reserved Event event (0x4029) **0** The common event is not implemented, or not counted.	
[8]	IDhi8	IDhi8 corresponds to a Reserved Event event (0x4028) **0** The common event is not implemented, or not counted.	
[7]	IDhi7	IDhi7 corresponds to a Reserved Event event (0x4027) **0** The common event is not implemented, or not counted.	
[6]	IDhi6	IDhi6 corresponds to common event (0x4026) MEM_ACCESS_CHECKED_WR **1** The common event is implemented.	
[5]	IDhi5	IDhi5 corresponds to common event (0x4025) MEM_ACCESS_CHECKED_RD **1** The common event is implemented.	
[4]	IDhi4	IDhi4 corresponds to common event (0x4024) MEM_ACCESS_CHECKED **1** The common event is implemented.	
[3]	IDhi3	IDhi3 corresponds to common event (0x4023) Reserved **0** The common event is not implemented, or not counted.	
[2]	IDhi2	IDhi2 corresponds to common event (0x4022) ST_ALIGN_LAT **1** The common event is implemented.	
[1]	IDhi1	IDhi1 corresponds to common event (0x4021) LD_ALIGN_LAT **1** The common event is implemented.	
[0]	IDhi0	IDhi0 corresponds to common event (0x4020) LDST_ALIGN_LAT **1** The common event is implemented.	

C.2.20 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 C-22: ext_pmmir bit assignments

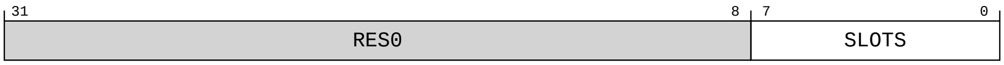


Table C-24: PMMIR bit descriptions

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. **`00000011`** The largest value by which the STALL_SLOT PMU event may increment in one cycle is 3.	

C.2.21 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 C-23: ext_pmdevarch bit assignments

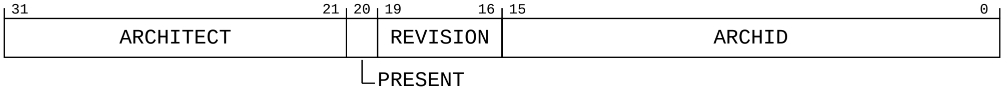


Table C-25: PMDEVARCH bit descriptions

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.	
[15:0]	ARCHID	Defines this part to be an Armv8 debug component. For architectures defined by Arm this is further subdivided. For Performance Monitors: • Bits [15:12] are the architecture version, 0x2. • Bits [11:0] are the architecture part number, 0xA16. This corresponds to Performance Monitors architecture version PMUv3. **Note:** The PMUv3 memory-mapped programmers' model can be used by devices other than Armv8 processors. Software must determine whether the PMU is attached to an Armv8 processor by using the ext-PMDEVAFF0 and ext-PMDEVAFF1 registers to discover the affinity of the PMU to any Armv8 processors.	

C.2.22 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 C-24: ext_pmdevid bit assignments

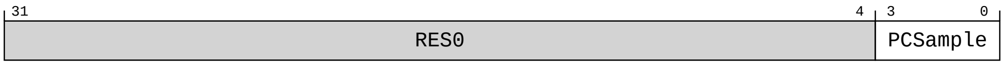


Table C-26: PMDEVID bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	`0b0`
[3:0]	PCSample	Indicates the level of PC Sample-based Profiling support using Performance Monitors registers. Permitted values of this field are: **`0001`** PC Sample-based Profiling Extension is implemented in the Performance Monitors register space.	

C.2.23 PMDEVTYPE, Performance Monitors Device Type register

Indicates to a debugger that this component is part of a PEs 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 C-25: ext_pmdevtype bit assignments

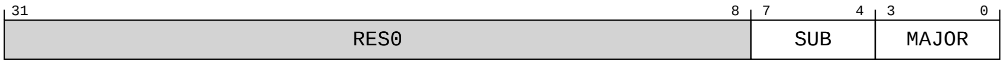


Table C-27: PMDEVTYPE bit descriptions

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.	

C.2.24 PMPIDR4, Performance Monitors Peripheral Identification Register 4

Provides information to identify a Performance Monitor component.

For more information see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-26: ext_pmpidr4 bit assignments

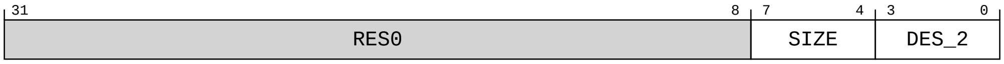


Table C-28: PMPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:4]	SIZE	Size of the component. RAZ. Log_2 of the number of 4KB pages from the start of the component to the end of the component ID registers.	
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. **`0100`** Arm Limited	

C.2.25 PMPIDR0, Performance Monitors Peripheral Identification Register 0

Provides information to identify a Performance Monitor component.

For more information see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-27: ext_pmpidr0 bit assignments

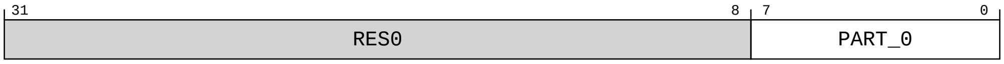


Table C-29: PMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`

Bits	Name	Description	Reset
[7:0]	PART_0	Part number, least significant byte. **01000110** Cortex®-A510	

C.2.26 PMPIDR1, Performance Monitors Peripheral Identification Register 1

Provides information to identify a Performance Monitor component.

For more information see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-28: ext_pmpidr1 bit assignments

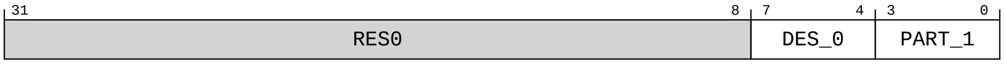


Table C-30: PMPIDR1 bit descriptions

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. **1011** Arm Limited	
[3:0]	PART_1	Part number, most significant nibble. **1101** Cortex®-A510	

C.2.27 PMPIDR2, Performance Monitors Peripheral Identification Register 2

Provides information to identify a Performance Monitor component.

For more information see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-29: ext_pmpidr2 bit assignments

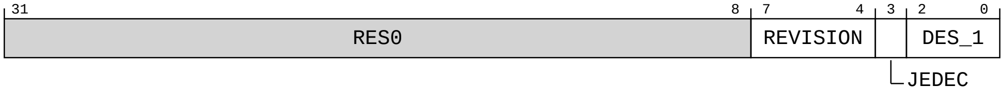


Table C-31: PMPIDR2 bit descriptions

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. **0000** r0p3	
[3]	JEDEC	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. **011** Arm Limited	

C.2.28 PMPIDR3, Performance Monitors Peripheral Identification Register 3

Provides information to identify a Performance Monitor component.

For more information see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-30: ext_pmpidr3 bit assignments

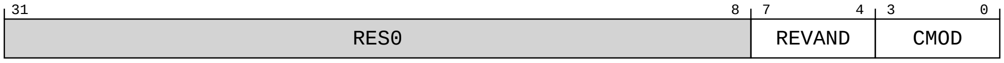


Table C-32: PMPIDR3 bit descriptions

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. **`0011`** r0p3	
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **`0000`**	

C.2.29 PMCIDR0, Performance Monitors Component Identification Register 0

Provides information to identify a Performance Monitor component.

For more information see *About the Component identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-31: ext_pmcidr0 bit assignments

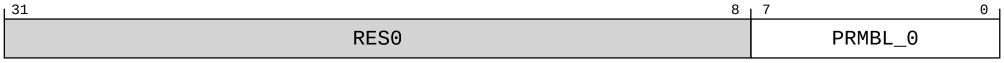


Table C-33: PMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:0]	PRMBL_0	Preamble. Must read as 0x0D.	

C.2.30 PMCIDR1, Performance Monitors Component Identification Register 1

Provides information to identify a Performance Monitor component.

For more information see *About the Component identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-32: ext_pmcidr1 bit assignments

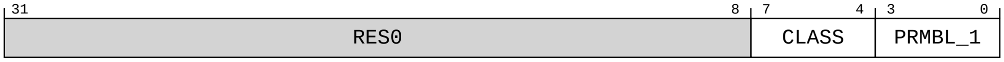


Table C-34: PMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	CLASS	Component class. Reads as 0x9, debug component.	
[3:0]	PRMBL_1	Preamble. RAZ.	

C.2.31 PMCIDR2, Performance Monitors Component Identification Register 2

Provides information to identify a Performance Monitor component.

For more information see *About the Component identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-33: ext_pmcidr2 bit assignments

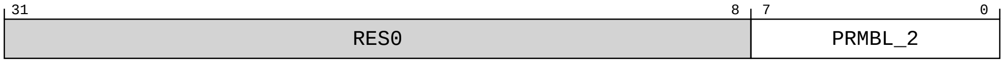


Table C-35: PMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_2	Preamble. Must read as 0x05.	

C.2.32 PMCIDR3, Performance Monitors Component Identification Register 3

Provides information to identify a Performance Monitor component.

For more information see *About the Component identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-34: ext_pmcidr3 bit assignments

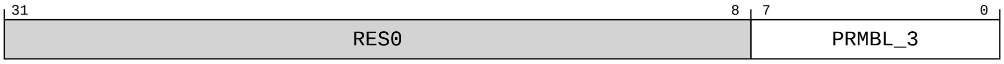


Table C-36: PMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:0]	PRMBL_3	Preamble. Must read as 0xB1.	

C.3 Debug register summary

The summary table provides an overview of memory-mapped debug registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table C-37: 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

C.3.1 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 C-35: ext_edrcr bit assignments

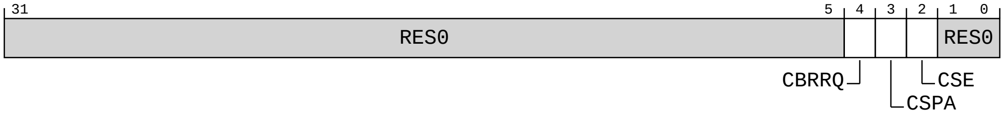


Table C-38: EDRCR bit descriptions

Bits	Name	Description	Reset
[31:5]	RES0	Reserved	`0b0`
[4]	CBRRQ	Allow imprecise entry to Debug state. The actions on writing to this bit are: **0** No action. **1** Allow imprecise entry to Debug state, for example by canceling pending bus accesses. Setting this bit to 1 allows a debugger to request imprecise entry to Debug state. An External Debug Request debug event must be pending before the debugger sets this bit to 1. This feature is optional. If this feature is not implemented, writes to this bit are ignored.	
[3]	CSPA	Clear Sticky Pipeline Advance. This bit is used to clear the ext-EDSCR.PipeAdv bit to 0. The actions on writing to this bit are: **0** No action. **1** Clear the ext-EDSCR.PipeAdv bit to 0.	
[2]	CSE	Clear Sticky Error. Used to clear the ext-EDSCR cumulative error bits to 0. The actions on writing to this bit are: **0** No action. **1** 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`

C.3.2 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 C-36: ext_edacr bit assignments

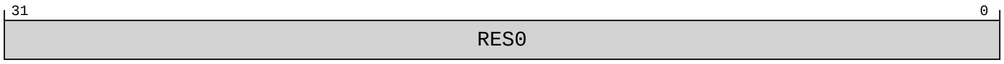


Table C-39: EDACR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.3.3 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 C-37: ext_edprcr bit assignments

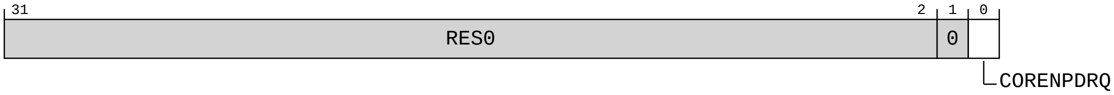


Table C-40: EDPRCR bit descriptions

Bits	Name	Description	Reset
[31:1]	RES0	Reserved	`0b0`
[0]	CORENPDRQ	Core no powerdown request. Requests emulation of powerdown. This request is typically passed to an external power controller. This means that whether a request causes power up is dependent on the **IMPLEMENTATION DEFINED** nature of the system. The power controller must not allow the Core power domain to switch off while this bit is 1. **0** If the system responds to a powerdown request, it powers down Core power domain. **1** If the system responds to a powerdown request, it does not powerdown the Core power domain, but instead emulates a powerdown of that domain. When this bit reads as UNKNOWN, the PE ignores writes to this bit. This field is in the Core power domain, and permitted accesses to this field map to the AArch32-DBG-PRCR.CORENPDRQ and AArch64-DBGPRCR_EL1.CORENPDRQ fields. In an implementation that includes the recommended external debug interface, this bit drives the DBGNOPWRDWN signal. It is **IMPLEMENTATION DEFINED** whether this bit is reset the Cold reset value on exit from an **IMPLEMENTATION DEFINED** software-visible retention state. For more information about retention states see Core power domain power states. **Note:** Writes to this bit are not prohibited by the **IMPLEMENTATION DEFINED** authentication interface. This means that a debugger can request emulation of powerdown regardless of whether invasive debug is permitted.	

C.3.4 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 C-38: ext_midr_el1 bit assignments

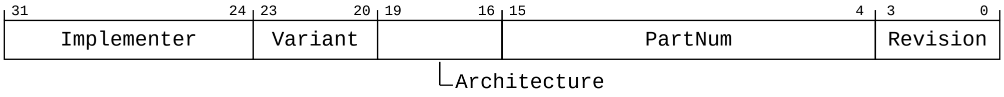


Table C-41: MIDR_EL1 bit descriptions

Bits	Name	Description	Reset
[31:24]	Implementer	Indicates the implementer code. This value is: **`01000001`** Arm Limited	
[23:20]	Variant	Indicates the major revision of the product. **`0000`** r0p3	
[19:16]	Architecture	The permitted values of this field are: **`1111`** Architecture is defined by ID registers	
[15:4]	PartNum	An **IMPLEMENTATION DEFINED** primary part number for the device. On processors implemented by Arm, if the top four bits of the primary part number are 0x0 or 0x7, the variant and architecture are encoded differently. **`110101000110`** Cortex®-A510	

Bits	Name	Description	Reset
[3:0]	Revision	Indicates the minor revision of the product. `0011` r0p3	

C.3.5 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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-39: ext_edpfr bit assignments

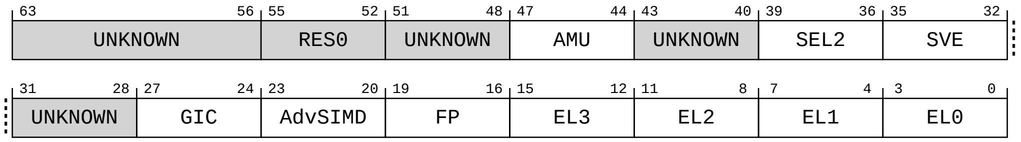


Table C-42: EDPFR bit descriptions

Bits	Name	Description	Reset
[63:56]	UNKNOWN	Reserved	
[55:52]	RES0	Reserved	`0b0`
[51:48]	UNKNOWN	Reserved	

Bits	Name	Description	Reset
[47:44]	AMU	Activity Monitors Extension. Defined values are: **0001** Activity Monitors Extension version 1 is implemented.	
[43:40]	UNKNOWN	Reserved	
[39:36]	SEL2	Secure EL2. Defined values are: **0001** Secure EL2 is implemented.	
[35:32]	SVE	Scalable Vector Extension. Defined values are: **0001** SVE is implemented.	
[31:28]	UNKNOWN	Reserved	
[27:24]	GIC	System register GIC interface support. Defined values are: **0011** System register interface to version 4.1 of the GIC CPU interface is supported.	
[23:20]	AdvSIMD	Advanced SIMD. Defined values are: **0001** Advanced SIMD is implemented, including support for the following SISD and SIMD operations: * Integer byte, halfword, word and doubleword element operations. * Half-precision, single-precision and double-precision floating-point arithmetic. * Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.	
[19:16]	FP	Floating-point. Defined values are: **0001** Floating-point is implemented, and includes support for: * Half-precision, single-precision and double-precision floating-point types. * Conversions between single-precision and half-precision data types, and double-precision and half-precision data types.	
[15:12]	EL3	AArch64 EL3 Exception level handling. Defined values are: **0001** EL3 can be executed in AArch64 state only.	
[11:8]	EL2	AArch64 EL2 Exception level handling. Defined values are: **0001** EL2 can be executed in AArch64 state only.	
[7:4]	EL1	AArch64 EL1 Exception level handling. Defined values are: **0001** EL1 can be executed in AArch64 state only.	
[3:0]	EL0	AArch64 EL0 Exception level handling. Defined values are: **0001** EL0 can be executed in AArch64 state only.	

C.3.6 EDDFR, External Debug Feature Register

Provides top level information about the debug system.

Note

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* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-40: ext_eddfr bit assignments

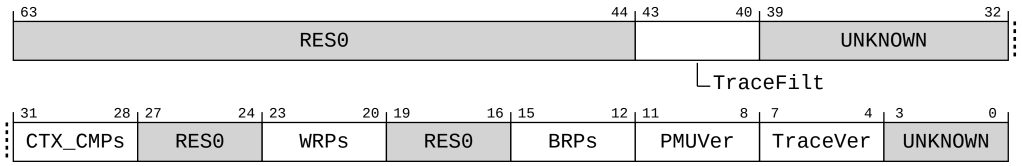


Table C-43: EDDFR bit descriptions

Bits	Name	Description	Reset
[63:44]	RES0	Reserved	0b0
[43:40]	TraceFilt	Armv8.4 Self-hosted Trace Extension version. Defined values are: `0001` Armv8.4 Self-hosted Trace Extension is implemented.	
[39:32]	UNKNOWN	Reserved	

Bits	Name	Description	Reset
[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. **`0001`** Two context-aware breakpoints are included	
[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. **`0011`** Four watchpoints	
[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. **`0101`** Six breakpoints	
[11:8]	PMUVer	Performance Monitors Extension version. Defined value is: **`0110`** 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: **`0001`** PE trace unit System registers implemented.	
[3:0]	UNKNOWN	Reserved	

C.3.7 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 C-41: ext_eddevarch bit assignments

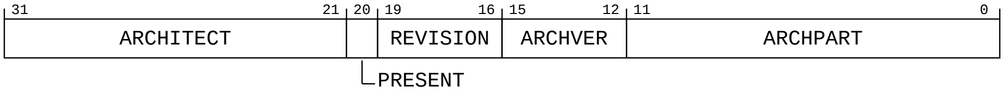


Table C-44: EDDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For debug, this is Arm Limited. Bits [31:28] are the JEP106 continuation code, 0x4. Bits [27:21] are the JEP106 ID code, 0x3B.	
[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 debug, the revision defined by Armv8-A is 0x0. All other values are reserved.	
[15:12]	ARCHVER	Defines the architecture version of the component. This is the same value as AArch64-ID_AA64DFR0_EL1.DebugVer and AArch32-DBGDIDR.Version. The defined values of this field are: **1001** Armv8.4 Debug architecture.	
[11:0]	ARCHPART	**101000010101** The part number of the Armv8-A debug component. The fields ARCHVER and ARCHPART together form the field ARCHID, so that ARCHPART is ARCHID[11:0].	

C.3.8 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 C-42: ext_eddevid2 bit assignments

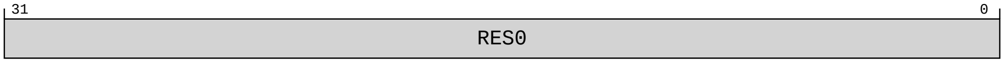


Table C-45: EDDEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

C.3.9 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 C-43: ext_eddevid1 bit assignments

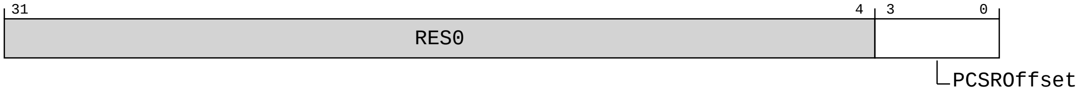


Table C-46: EDDEVID1 bit descriptions

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. Permitted values of this field in Armv8 are: `0000` ext-EDPCSR not implemented.	

C.3.10 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 C-44: ext_eddevid bit assignments

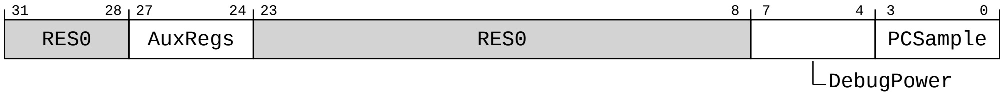


Table C-47: EDDEVID bit descriptions

Bits	Name	Description	Reset
[31:28]	RES0	Reserved	0b0
[27:24]	AuxRegs	Indicates support for Auxiliary registers. Permitted values for this field are: `0000` None supported.	
[23:8]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[7:4]	DebugPower	Indicates support for the ARMv8.3-DoPD feature. Defined values of this field are: `0001` 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. Permitted values of this field are: `0000` PC Sample-based Profiling Extension is not implemented in the external debug registers space.	

C.3.11 EDDEVTYPE, External Debug Device Type register

Indicates to a debugger that this component is part of a PEs 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 C-45: ext_eddevtype bit assignments

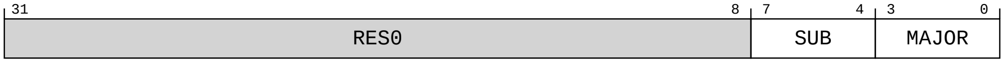


Table C-48: EDDEVTYPE bit descriptions

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.	

C.3.12 EDPIDR4, External Debug Peripheral Identification Register 4

Provides information to identify an external debug component.

For more information see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-46: ext_edpidr4 bit assignments

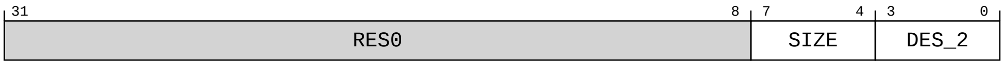


Table C-49: EDPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:4]	SIZE	Size of the component. RAZ. Log_2 of the number of 4KB pages from the start of the component to the end of the component ID registers.	
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. **`0100`** Arm Limited	

C.3.13 EDPIDR0, External Debug Peripheral Identification Register 0

Provides information to identify an external debug component.

For more information see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-47: ext_edpidr0 bit assignments

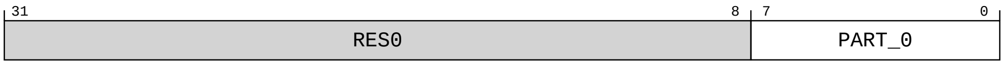


Table C-50: EDPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:0]	PART_0	Part number, least significant byte. **`01000110`** Cortex®-A510	

C.3.14 EDPIDR1, External Debug Peripheral Identification Register 1

Provides information to identify an external debug component.

For more information see *About the Peripheral identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-48: ext_edpidr1 bit assignments

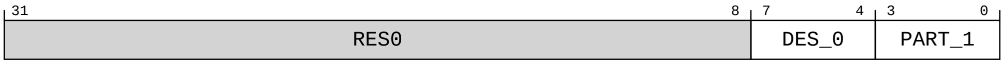


Table C-51: EDPIDR1 bit descriptions

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. **`1011`** Arm Limited	
[3:0]	PART_1	Part number, most significant nibble. **`1101`** Cortex®-A510	

C.3.15 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 C-49: ext_edpidr2 bit assignments

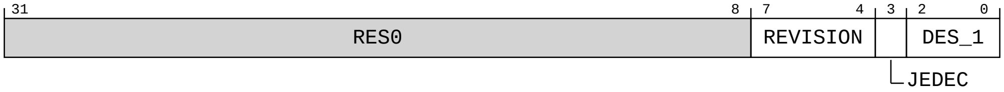


Table C-52: EDPIDR2 bit descriptions

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. **0000** r0p3	
[3]	JEDEC	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. **011** Arm Limited	

C.3.16 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 C-50: ext_edpidr3 bit assignments

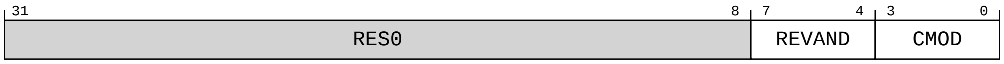


Table C-53: EDPIDR3 bit descriptions

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. **`0011`** r0p3	
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. **`0000`**	

C.3.17 EDCIDR0, External Debug Component Identification Register 0

Provides information to identify an external debug component.

For more information see *About the Component identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-51: ext_edcidr0 bit assignments

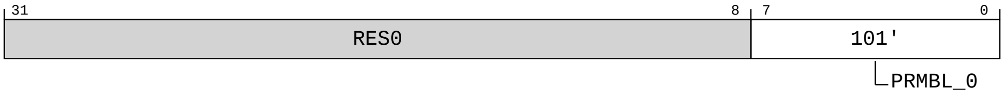


Table C-54: EDCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_0	Preamble.	0b1101

C.3.18 EDCIDR1, External Debug Component Identification Register 1

Provides information to identify an external debug component.

For more information see *About the Component identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-52: ext_edcidr1 bit assignments

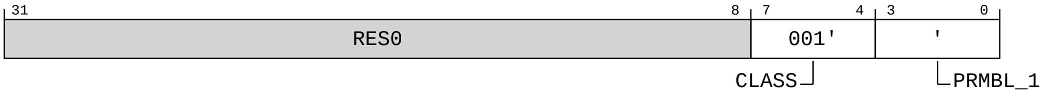


Table C-55: EDCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	CLASS	Component class. Debug component.	0b1001
[3:0]	PRMBL_1	Preamble.	0b0

C.3.19 EDCIDR2, External Debug Component Identification Register 2

Provides information to identify an external debug component.

For more information see *About the Component identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-53: ext_edcidr2 bit assignments

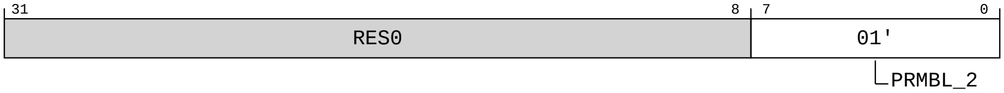


Table C-56: EDCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_2	Preamble.	0b101

C.3.20 EDCIDR3, External Debug Component Identification Register 3

Provides information to identify an external debug component.

For more information see *About the Component identification scheme* in the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile*.

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 C-54: ext_edcidr3 bit assignments

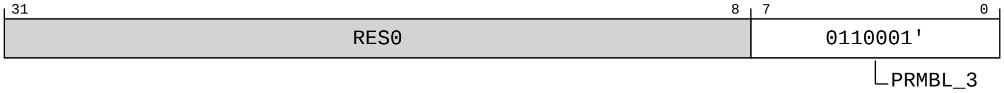


Table C-57: EDCIDR3 bit descriptions

Bits	Name	Description	
[31:8]	RES0	Reserved	`0b0`
[7:0]	PRMBL_3	Preamble.	`0b10110001`

C.4 Memory-mapped AMU register summary

The summary table provides an overview of memory-mapped *Activity Monitoring Unit* (AMU) registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table C-58: 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

Offset	Name	Reset	Width	Description
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
0xCE0	AMCGCR	See individual bit resets	32-bit	Activity Monitors Counter Group Configuration Register
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

C.4.1 AMEVTYPER00, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR00_EL0 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 C-55: ext_amevtyper00 bit assignments

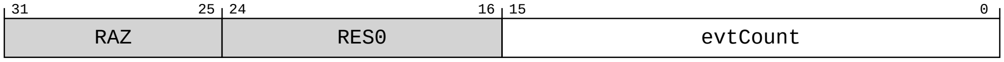


Table C-59: AMEVTYPER00 bit descriptions

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 mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **`0000000000010001`** Processor frequency cycles	

C.4.2 AMEVTYPER01, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR01_EL0 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 C-56: ext_amevtyper01 bit assignments

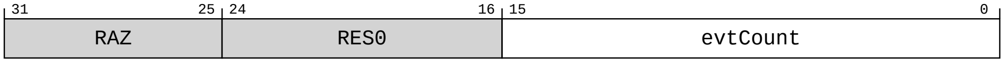


Table C-60: AMEVTYPER01 bit descriptions

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 mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: **0100000000000100** Constant frequency cycles	

C.4.3 AMEVTYPER02, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR02_EL0 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 C-57: ext_amevtyper02 bit assignments

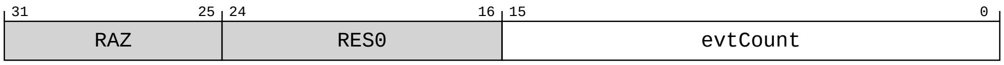


Table C-61: AMEVTYPER02 bit descriptions

Bits	Name	Description	Reset
[31:25]	RAZ	Reserved	
[24:16]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[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 mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: `0000000000001000` Instructions retired	

C.4.4 AMEVTYPER03, Activity Monitors Event Type Registers 0

Provides information on the events that an architected activity monitor event counter AArch64-AMEVCNTR03_EL0 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 C-58: ext_amevtyper03 bit assignments

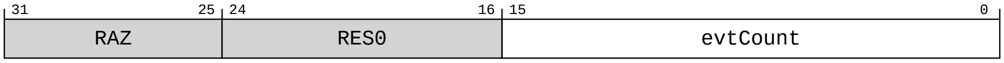


Table C-62: AMEVTYPER03 bit descriptions

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 mandated for each architected counter. The following table shows the mapping between required event numbers and the corresponding counters: `0100000000000101` Memory stall cycles	

C.4.5 AMEVTYPER10, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AArch64-AMEVCNTR10_EL0 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 C-59: ext_amevtyper10 bit assignments

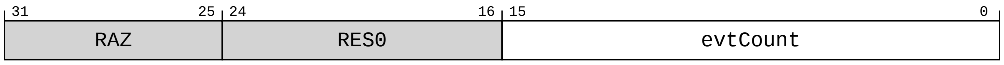


Table C-63: AMEVTYPER10 bit descriptions

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 auxiliary activity monitor event counter AMEVCNTR10_EL0. **`0000001100000000`** MPMM gear 0 period threshold exceeded	

C.4.6 AMEVTYPER11, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AArch64-AMEVCNTR11_EL0 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 C-60: ext_amevtyper11 bit assignments

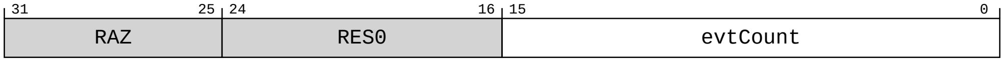


Table C-64: AMEVTYPER11 bit descriptions

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 auxiliary activity monitor event counter AMEVCNTR11_EL0. **`0000001100000001`** MPMM gear 1 period threshold exceeded	

C.4.7 AMEVTYPER12, Activity Monitors Event Type Registers 1

Provides information on the events that an auxiliary activity monitor event counter AArch64-AMEVCNTR12_EL0 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 C-61: ext_amevtyper12 bit assignments

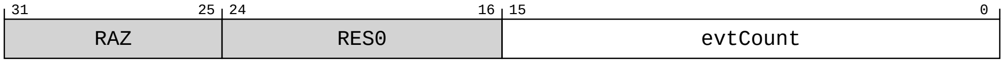


Table C-65: AMEVTYPER12 bit descriptions

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 auxiliary activity monitor event counter AMEVCNTR12_EL0. **`0000001100000010`** MPMM gear 2 period threshold exceeded	

C.4.8 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 C-62: ext_amcgcr bit assignments

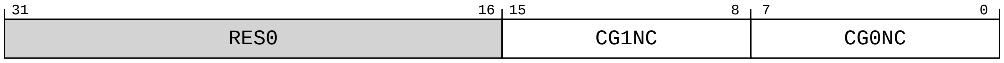


Table C-66: AMCGCR bit descriptions

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. **00000011** 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. **00000100** Four counters in the architected counter group	

C.4.9 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 C-63: ext_amcfgr bit assignments

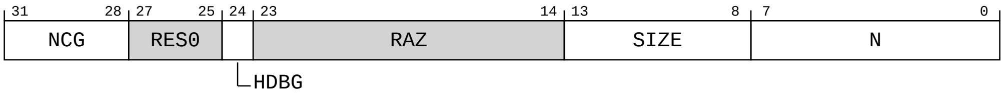


Table C-67: AMCFGR bit descriptions

Bits	Name	Description	Reset
[31:28]	NCG	Defines the number of counter groups. The following value is specified for this product. **0001** 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. **1** 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. **Note:** Software also uses this field to determine the spacing of counters in the memory-map. From Armv8, the counters are at doubleword-aligned addresses. **111111** 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]. **00000110** Seven activity monitor event counters	

C.4.10 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 C-64: ext_amiidr bit assignments

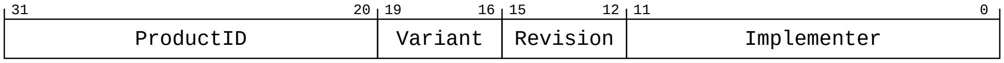


Table C-68: AMIIDR bit descriptions

Bits	Name	Description	Reset
[31:20]	ProductID	This field is an AMU part identifier. The value of this field is **IMPLEMENTATION DEFINED**. **110101000110** Cortex®-A510	
[19:16]	Variant	This field distinguishes product variants or major revisions of the product. The value of this field is **IMPLEMENTATION DEFINED**. **0000** r0p3	
[15:12]	Revision	This field distinguishes minor revisions of the product. The value of this field is **IMPLEMENTATION DEFINED**. **0011** r0p3	
[11:0]	Implementer	Contains the JEP106 code of the company that implemented the AMU. For an Arm implementation, this field reads as 0x43B. **010000111011** Arm Limited	

C.4.11 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 C-65: ext_amdevarch bit assignments

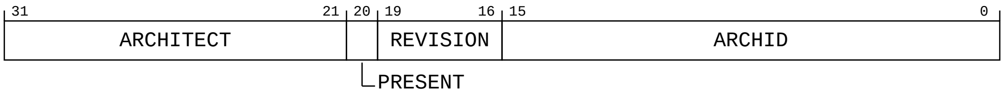


Table C-69: AMDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Defines the architecture of the component. For AMU, 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. `0000` Architecture revision is AMUv1. All other values are reserved.	
[15:0]	ARCHID	Defines this part to be an AMU component. For architectures defined by Arm this is further subdivided. For AMU: • Bits [15:12] are the architecture version, 0x0. • Bits [11:0] are the architecture part number, 0xA66. This corresponds to AMU architecture version AMUv1.	

C.4.12 AMDEVTYPE, 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 C-66: ext_amdevtype bit assignments

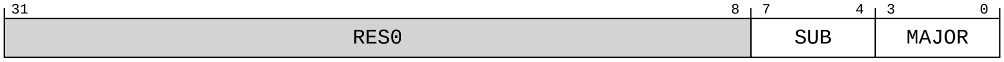


Table C-70: AMDEVTYPE bit descriptions

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.	

C.4.13 AMPIDR4, Activity Monitors Peripheral Identification Register 4

Provides information to identify an activity monitors component.

For more information, see About the Peripheral identification scheme in the Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile.

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 C-67: ext_ampidr4 bit assignments

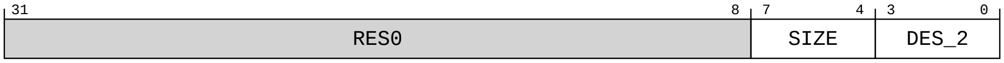


Table C-71: AMPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	SIZE	Size of the component. Log_2 of the number of 4KB pages from the start of the component to the end of the component ID registers. This field reads as 0b0000.	
[3:0]	DES_2	Designer. JEP106 continuation code, least significant nibble. The value of this field is **IMPLEMENTATION DEFINED**. For Arm Limited, this field is 0b0100. **0100** Arm Limited	

C.4.14 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 C-68: ext_ampidr0 bit assignments

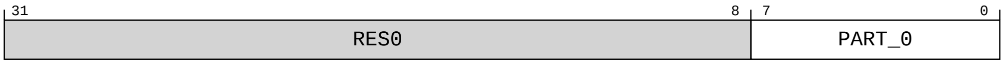


Table C-72: AMPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:0]	PART_0	Part number, least significant byte. The value of this field is IMPLEMENTATION DEFINED. **`01000110`** Cortex®-A510	

C.4.15 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 C-69: ext_ampidr1 bit assignments

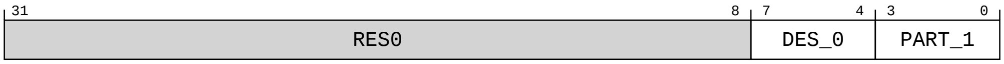


Table C-73: AMPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:4]	DES_0	Designer, least significant nibble of JEP106 ID code. The value of this field is IMPLEMENTATION DEFINED. For Arm Limited, this field is 0b1011. **`1011`** Arm Limited	
[3:0]	PART_1	Part number, most significant nibble. The value of this field is IMPLEMENTATION DEFINED. **`1101`** Cortex®-A510	

C.4.16 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 C-70: ext_ampidr2 bit assignments

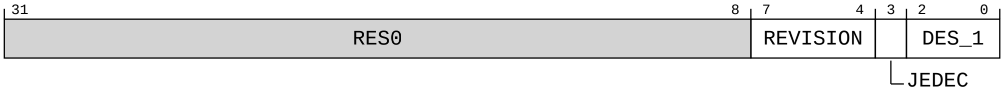


Table C-74: AMPIDR2 bit descriptions

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 **IMPLEMENTATION DEFINED**. **`0000`** r0p3	
[3]	JEDEC	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 **IMPLEMENTATION DEFINED**. For Arm Limited, this field is 0b011. **`011`** Arm Limited	

C.4.17 AMPIDR3, Activity Monitors Peripheral Identification Register 3

Provides information to identify an activity monitors component.

For more information, see About the Peripheral identification scheme in the Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile.

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 C-71: ext_ampidr3 bit assignments

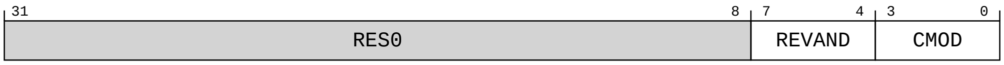


Table C-75: AMPIDR3 bit descriptions

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 IMPLEMENTATION DEFINED. **`0011`** r0p3	
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. The value of this field is IMPLEMENTATION DEFINED. **`0000`**	

C.4.18 AMCIDR0, Activity Monitors Component Identification Register 0

Provides information to identify an activity monitors component.

For more information, see About the Component identification scheme in the Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile.

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 C-72: ext_amcidr0 bit assignments

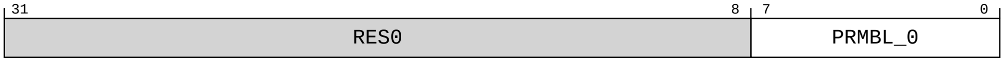


Table C-76: AMCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:0]	PRMBL_0	Preamble. Must read as 0x0D.	

C.4.19 AMCIDR1, Activity Monitors Component Identification Register 1

Provides information to identify an activity monitors component.

For more information, see About the Component identification scheme in the Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile.

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 C-73: ext_amcidr1 bit assignments

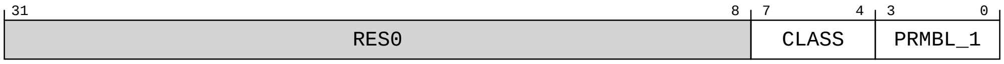


Table C-77: AMCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:4]	CLASS	Component class. Reads as 0x9, CoreSight component.	
[3:0]	PRMBL_1	Preamble. Reads as 0x0.	

C.4.20 AMCIDR2, Activity Monitors Component Identification Register 2

Provides information to identify an activity monitors component.

For more information, see About the Component identification scheme in the Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile.

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 C-74: ext_amcidr2 bit assignments

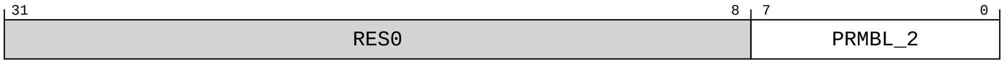


Table C-78: AMCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_2	Preamble. Reads as 0x05.	

C.4.21 AMCIDR3, Activity Monitors Component Identification Register 3

Provides information to identify an activity monitors component.

For more information, see About the Component identification scheme in the Arm® Architecture Reference Manual, Armv8, for Armv8-A architecture profile.

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 C-75: ext_amcidr3 bit assignments

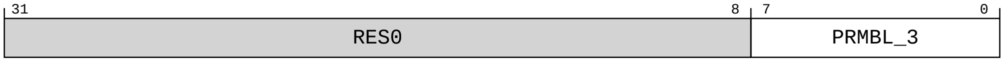


Table C-79: AMCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:0]	PRMBL_3	Preamble. Reads as 0xB1.	

C.5 RAS register summary

The summary table provides an overview of *Reliability, Availability, Serviceability* (RAS) registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table C-80: RAS register summary

Name	Reset	Width	Description
ERR2CTLR	See individual bit resets	64-bit	Error Record Control Register
ERR1CTLR	See individual bit resets	64-bit	Error Record Control Register
ERR2FR	See individual bit resets	64-bit	Error Record Feature Register
ERR1FR	See individual bit resets	64-bit	Error Record Feature Register
ERR2MISC1	`0x0`	64-bit	Error Record Miscellaneous Register 1
ERR1MISC1	`0x0`	64-bit	Error Record Miscellaneous Register 1
ERR2MISC0	See individual bit resets	64-bit	Error Record Miscellaneous Register 0
ERR1MISC0	See individual bit resets	64-bit	Error Record Miscellaneous Register 0
ERR2PFGF	See individual bit resets	64-bit	Pseudo-fault Generation Feature Register
ERR1PFGF	See individual bit resets	64-bit	Pseudo-fault Generation Feature Register
ERR2PFGCTL	See individual bit resets	64-bit	Pseudo-fault Generation Control Register

Name	Reset	Width	Description
ERR1PFGCTL	See individual bit resets	64-bit	Pseudo-fault Generation Control Register
ERR2STATUS	See individual bit resets	64-bit	Error Record Primary Status Register
ERR1STATUS	See individual bit resets	64-bit	Error Record Primary Status Register
ERR2MISC3	0x0	64-bit	Error Record Miscellaneous Register 3
ERR1MISC3	0x0	64-bit	Error Record Miscellaneous Register 3
ERR2MISC2	0x0	64-bit	Error Record Miscellaneous Register 2
ERR1MISC2	0x0	64-bit	Error Record Miscellaneous Register 2

C.5.1 ERR2CTLR, Error Record Control Register

The error control register contains enable bits for the node that writes to this record:

- Enabling error detection and correction.
- Enabling the critical error, error recovery, and fault handling interrupts.
- Enabling in-band error response for Uncorrected errors.

For each bit, if the selected node does not support the feature, then the bit is RES0. The definition of each record is **IMPLEMENTATION DEFINED**.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure C-76: ext_err2ctlr bit assignments

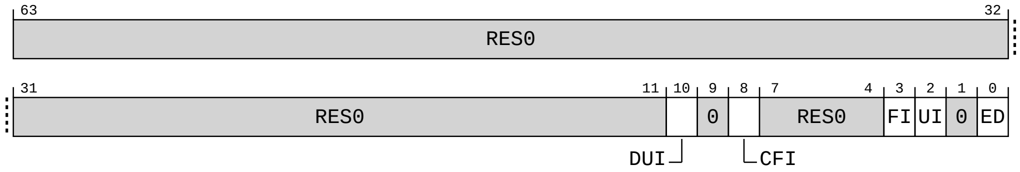


Table C-81: ERR2CTLR bit descriptions

Bits	Name	Description	Reset
[63:11]	RES0	Reserved	`0b0`
[10]	DUI	Error recovery interrupt for deferred errors enable. When ext-ERR<n>FR.DUI == 0b10, this control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all detected Deferred errors. **0** Error recovery interrupt not generated for deferred errors. **1** Error recovery interrupt generated for deferred errors. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	
[9]	RES0	Reserved	`0b0`
[8]	CFI	Fault handling interrupt for Corrected errors enable. When ext-ERR<n>FR.CFI == 0b10, this control applies to errors arising from both reads and writes. When enabled: • If the node implements Corrected error counters, then the fault handling interrupt is generated when a counter overflows and the overflow bit for the counter is set to 0b1. For more information, see ext-ERR<n>MISC0. • Otherwise, the fault handling interrupt is also generated for all detected Corrected errors. **0** Fault handling interrupt not generated for Corrected errors. **1** Fault handling interrupt generated for Corrected errors. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	
[7:4]	RES0	Reserved	`0b0`

Bits	Name	Description	Reset
[3]	FI	Fault handling interrupt enable. When ext-ERR<n>FR.FI == 0b10, this control applies to errors arising from both reads and writes. When enabled: • The fault handling interrupt is generated for all detected Deferred errors and Uncorrected errors. • If the fault handling interrupt for Corrected errors control is not implemented: ◦ If the node implements Corrected error counters, then the fault handling interrupt is also generated when a counter overflows and the overflow bit for the counter is set to 0b1. ◦ Otherwise, the fault handling interrupt is also generated for all detected Corrected errors. **0** Fault handling interrupt disabled. **1** Fault handling interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	
[2]	UI	Uncorrected error recovery interrupt enable. When ext-ERR<n>FR.UI == 0b10, this control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all detected Uncorrected errors that are not deferred. **0** Error recovery interrupt disabled. **1** Error recovery interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	
[1]	RES0	Reserved	`0b0`
[0]	ED	Error reporting and logging enable. When disabled, the node behaves as if error detection and correction are disabled, and no errors are recorded or signaled by the node. Arm recommends that, when disabled, correct error detection and correction codes are written for writes, unless disabled by an IMPLEMENTATION DEFINED control for error injection. **0** Error reporting disabled. **1** Error reporting enabled. It is IMPLEMENTATION DEFINED whether the node fully disables error detection and correction when reporting is disabled. That is, even with error reporting disabled, the node might continue to silently correct errors. Uncorrectable errors might result in corrupt data being silently propagated by the node. **Note:** If this node requires initialization after Cold reset to prevent signaling false errors, then Arm recommends this bit is set to 0b0 on Cold reset, meaning errors are not reported from Cold reset. This allows boot software to initialize a node without signaling errors. Software can enable error reporting after the node is initialized. Otherwise, the Cold reset value is IMPLEMENTATION DEFINED. If the Cold reset value is 0b1, the reset values of other controls in this register are also IMPLEMENTATION DEFINED and should not be UNKNOWN.	

C.5.2 ERR1CTLR, Error Record Control Register

The error control register contains enable bits for the node that writes to this record:

- Enabling error detection and correction.
- Enabling the critical error, error recovery, and fault handling interrupts.
- Enabling in-band error response for Uncorrected errors.

For each bit, if the selected node does not support the feature, then the bit is RES0. The definition of each record is **IMPLEMENTATION DEFINED**.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure C-77: ext_err1ctlr bit assignments

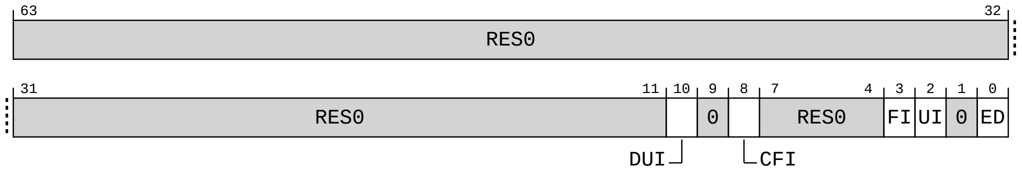


Table C-82: ERR1CTLR bit descriptions

Bits	Name	Description	Reset
[63:11]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[10]	DUI	Error recovery interrupt for deferred errors enable. When ext-ERR<n>FR.DUI == 0b10, this control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all detected Deferred errors. **0** Error recovery interrupt not generated for deferred errors. **1** Error recovery interrupt generated for deferred errors. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	
[9]	RES0	Reserved	`0b0`
[8]	CFI	Fault handling interrupt for Corrected errors enable. When ext-ERR<n>FR.CFI == 0b10, this control applies to errors arising from both reads and writes. When enabled: • If the node implements Corrected error counters, then the fault handling interrupt is generated when a counter overflows and the overflow bit for the counter is set to 0b1. For more information, see ext-ERR<n>MISC0. • Otherwise, the fault handling interrupt is also generated for all detected Corrected errors. **0** Fault handling interrupt not generated for Corrected errors. **1** Fault handling interrupt generated for Corrected errors. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	
[7:4]	RES0	Reserved	`0b0`
[3]	FI	Fault handling interrupt enable. When ext-ERR<n>FR.FI == 0b10, this control applies to errors arising from both reads and writes. When enabled: • The fault handling interrupt is generated for all detected Deferred errors and Uncorrected errors. • If the fault handling interrupt for Corrected errors control is not implemented: ◦ If the node implements Corrected error counters, then the fault handling interrupt is also generated when a counter overflows and the overflow bit for the counter is set to 0b1. ◦ Otherwise, the fault handling interrupt is also generated for all detected Corrected errors. **0** Fault handling interrupt disabled. **1** Fault handling interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	

Bits	Name	Description	Reset
[2]	UI	Uncorrected error recovery interrupt enable. When ext-ERR<n>FR.UI == 0b10, this control applies to errors arising from both reads and writes. When enabled, the error recovery interrupt is generated for all detected Uncorrected errors that are not deferred. **0** Error recovery interrupt disabled. **1** Error recovery interrupt enabled. The interrupt is generated even if the error syndrome is discarded because the error record already records a higher priority error.	
[1]	RES0	Reserved	`0b0`
[0]	ED	Error reporting and logging enable. When disabled, the node behaves as if error detection and correction are disabled, and no errors are recorded or signaled by the node. Arm recommends that, when disabled, correct error detection and correction codes are written for writes, unless disabled by an **IMPLEMENTATION DEFINED** control for error injection. **0** Error reporting disabled. **1** Error reporting enabled. It is **IMPLEMENTATION DEFINED** whether the node fully disables error detection and correction when reporting is disabled. That is, even with error reporting disabled, the node might continue to silently correct errors. Uncorrectable errors might result in corrupt data being silently propagated by the node. **Note:** If this node requires initialization after Cold reset to prevent signaling false errors, then Arm recommends this bit is set to 0b0 on Cold reset, meaning errors are not reported from Cold reset. This allows boot software to initialize a node without signaling errors. Software can enable error reporting after the node is initialized. Otherwise, the Cold reset value is **IMPLEMENTATION DEFINED**. If the Cold reset value is 0b1, the reset values of other controls in this register are also **IMPLEMENTATION DEFINED** and should not be UNKNOWN.	

C.5.3 ERR2FR, Error Record Feature Register

Defines whether <n> is the first record owned by a node:

- If <n> is the first error record owned by a node, then ERR<n>FR.ED != 0b00.
- If <n> is not the first error record owned by a node, then ERR<n>FR.ED == 0b00.

If <n> is the first record owned by the node, defines which of the common architecturally-defined features are implemented by the node and, of the implemented features, which are software programmable.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure C-78: ext_err2fr bit assignments

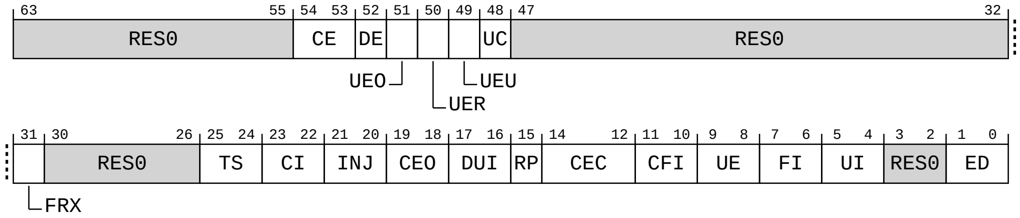


Table C-83: ERR2FR bit descriptions

Bits	Name	Description	Reset
[63:55]	RES0	Reserved	0b0
[54:53]	CE	Corrected Error recording. Describes the types of Corrected Error the node can record. **10** The node can record of a non-specific Corrected Error (a Corrected Error that is recorded as ext-ERR<n>STATUS.CE == 0b10).	
[52]	DE	Deferred Error recording. Describes whether the node can record this type of error. **1** The node can record this type of error.	
[51]	UEO	Latent or Restartable Error recording. Describes whether the node can record this type of error. **0** The node does not record this type of error.	
[50]	UER	Signaled or Recoverable Error recording. Describes whether the node can record this type of error. **0** The node does not record this type of error.	
[49]	UEU	Unrecoverable Error recording. Describes whether the node can record this type of error. **0** The node does not record this type of error.	
[48]	UC	Uncontainable Error recording. Describes whether the node can record this type of error. **1** The node can record this type of error.	
[47:32]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[31]	FRX	Feature Register extension. Defines whether ERR<n>FR[63:48] are architecturally defined. **1** ERR<n>FR[63:48] are defined by the architecture.	
[30:26]	RES0	Reserved	`0b0`
[25:24]	TS	Timestamp Extension. Indicates whether, for each error record <m> owned by this node, `ERR<m>MISC3` is used as the timestamp register, and, if it is, the timebase used by the timestamp. **00** 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. **00** Does not support the critical error interrupt. ext-ERR<n>CTLR.CI is RES0.	
[21:20]	INJ	Fault Injection Extension. Indicates whether the RAS Common Fault Injection Model Extension is implemented. **01** The node implements the RAS Common Fault Injection Model Extension. See ext-ERR<n>PFGF 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. **00** Counts Corrected errors if a counter is implemented. Keeps the previous error syndrome. If the counter overflows, or no counter is implemented, then `ERR<m>STATUS`.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. **10** Control for enabling error recovery interrupts on deferred errors is supported and controllable using ext-ERR<n>CTLR.DUI.	
[15]	RP	Repeat counter. Indicates whether the node implements the repeat Corrected error counter in `ERR<m>MISC0` for each error record <m> owned by the node that implements the standard Corrected error counter. **1** A first (repeat) counter and a second (other) counter are implemented. The repeat counter is the same size as the primary error counter.	
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter (CE counter) mechanisms in `ERR<m>MISC0` for each error record <m> owned by the node that can record countable errors. **010** Implements an 8-bit Corrected error counter in `ERR<m>MISC0`[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. **10** Control for enabling fault handling interrupts on corrected errors is supported and controllable using ext-ERR<n>CTLR.CFI.	
[9:8]	UE	In-band uncorrected error reporting. Indicates whether the in-band uncorrected error reporting (External Aborts) and associated controls are implemented. **01** In-band uncorrected error reporting (External Aborts) is supported and always enabled. ext-ERR<n>CTLR.UE is RES0.	

Bits	Name	Description	Reset
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented. **10** Fault handling interrupt is supported and controllable using ext-ERR<n>CTLR.FI.	
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented. **10** Error handling interrupt is supported and controllable using ext-ERR<n>CTLR.UI.	
[3:2]	RES0	Reserved	0b0
[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. **10** Error reporting and logging is controllable using ext-ERR<n>CTLR.ED.	

C.5.4 ERR1FR, Error Record Feature Register

Defines whether <n> is the first record owned by a node:

- If <n> is the first error record owned by a node, then ERR<n>FR.ED != 0b00.
- If <n> is not the first error record owned by a node, then ERR<n>FR.ED == 0b00.

If <n> is the first record owned by the node, defines which of the common architecturally-defined features are implemented by the node and, of the implemented features, which are software programmable.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure C-79: ext_err1fr bit assignments

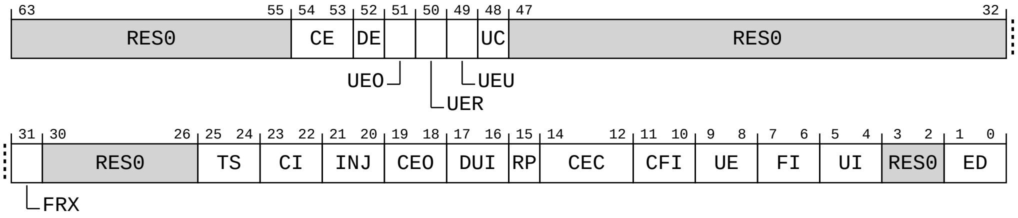


Table C-84: ERR1FR bit descriptions

Bits	Name	Description	Reset
[63:55]	RES0	Reserved	`0b0`
[54:53]	CE	Corrected Error recording. Describes the types of Corrected Error the node can record. **10** The node can record of a non-specific Corrected Error (a Corrected Error that is recorded as ext-ERR<n>STATUS.CE == 0b10).	
[52]	DE	Deferred Error recording. Describes whether the node can record this type of error. **1** The node can record this type of error.	
[51]	UEO	Latent or Restartable Error recording. Describes whether the node can record this type of error. **0** The node does not record this type of error.	
[50]	UER	Signaled or Recoverable Error recording. Describes whether the node can record this type of error. **0** The node does not record this type of error.	
[49]	UEU	Unrecoverable Error recording. Describes whether the node can record this type of error. **1** The node can record this type of error.	
[48]	UC	Uncontainable Error recording. Describes whether the node can record this type of error. **1** The node can record this type of error.	
[47:32]	RES0	Reserved	`0b0`
[31]	FRX	Feature Register extension. Defines whether ERR<n>FR[63:48] are architecturally defined. **1** ERR<n>FR[63:48] are defined by the architecture.	
[30:26]	RES0	Reserved	`0b0`
[25:24]	TS	Timestamp Extension. Indicates whether, for each error record <m> owned by this node, `ERR<m>MISC3` is used as the timestamp register, and, if it is, the timebase used by the timestamp. **00** The node does not support a timestamp register.	

Bits	Name	Description	Reset
[23:22]	CI	Critical error interrupt. Indicates whether the critical error interrupt and associated controls are implemented. **`00`** Does not support the critical error interrupt. ext-ERR<n>CTLR.CI is RES0.	
[21:20]	INJ	Fault Injection Extension. Indicates whether the RAS Common Fault Injection Model Extension is implemented. **`01`** The node implements the RAS Common Fault Injection Model Extension. See ext-ERR<n>PFGF 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. **`00`** Counts Corrected errors if a counter is implemented. Keeps the previous error syndrome. If the counter overflows, or no counter is implemented, then `ERR<m>STATUS`.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. **`10`** Control for enabling error recovery interrupts on deferred errors is supported and controllable using ext-ERR<n>CTLR.DUI.	
[15]	RP	Repeat counter. Indicates whether the node implements the repeat Corrected error counter in `ERR<m>MISC0` for each error record <m> owned by the node that implements the standard Corrected error counter. **`1`** A first (repeat) counter and a second (other) counter are implemented. The repeat counter is the same size as the primary error counter.	
[14:12]	CEC	Corrected Error Counter. Indicates whether the node implements the standard Corrected error counter (CE counter) mechanisms in `ERR<m>MISC0` for each error record <m> owned by the node that can record countable errors. **`010`** Implements an 8-bit Corrected error counter in `ERR<m>MISC0`[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. **`10`** Control for enabling fault handling interrupts on corrected errors is supported and controllable using ext-ERR<n>CTLR.CFI.	
[9:8]	UE	In-band uncorrected error reporting. Indicates whether the in-band uncorrected error reporting (External Aborts) and associated controls are implemented. **`01`** In-band uncorrected error reporting (External Aborts) is supported and always enabled. ext-ERR<n>CTLR.UE is RES0.	
[7:6]	FI	Fault handling interrupt. Indicates whether the fault handling interrupt and associated controls are implemented. **`10`** Fault handling interrupt is supported and controllable using ext-ERR<n>CTLR.FI.	
[5:4]	UI	Error recovery interrupt for uncorrected errors. Indicates whether the error handling interrupt and associated controls are implemented. **`10`** Error handling interrupt is supported and controllable using ext-ERR<n>CTLR.UI.	
[3:2]	RES0	Reserved	`0b0`

Bits	Name	Description	Reset
[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. `10` Error reporting and logging is controllable using ext-ERR<n>CTLR.ED.	

C.5.5 ERR2MISC1, Error Record Miscellaneous Register 1

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to identify the FRU in which the error was detected, and might contain enough information to locate the error within that FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

`0x0`

Bit descriptions

Figure C-80: ext_err2misc1 bit assignments

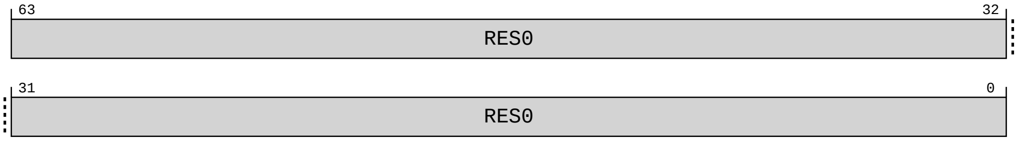


Table C-85: ERR2MISC1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	`0b0`

C.5.6 ERR1MISC1, Error Record Miscellaneous Register 1

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to identify the FRU in which the error was detected, and might contain enough information to locate the error within that FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

`0x0`

Bit descriptions

Figure C-81: ext_err1misc1 bit assignments

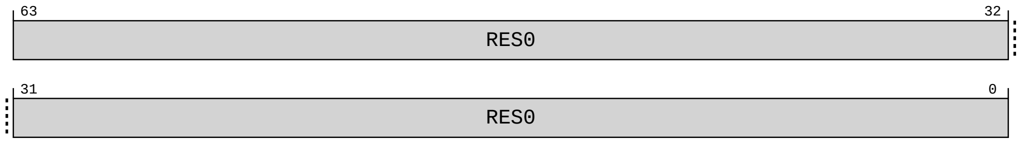


Table C-86: ERR1MISC1 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	`0b0`

C.5.7 ERR2MISC0, Error Record Miscellaneous Register 0

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to identify the FRU in which the error was detected, and might contain enough information to locate the error within that FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

If the node that owns error record <n> implements architecturally-defined error counters (ERR<q>FR.CEC != 0b000), and error record <n> can record countable errors, then ERR<n>MISC0 implements the architecturally-defined error counter or counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure C-82: ext_err2misc0 bit assignments

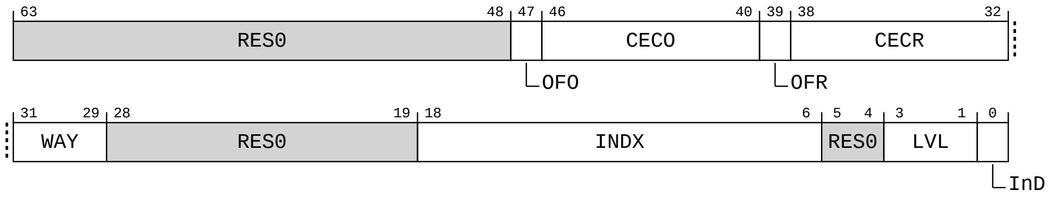


Table C-87: ERR2MISC0 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	`0b0`
[47]	OFO	Sticky overflow bit, other. Set to 1 when ERR<n>MISC0.CECO is incremented and wraps through zero. **0** Other counter has not overflowed. **1** Other counter has overflowed. A direct write that modifies this bit might indirectly set ext-ERR<n>STATUS.OF to an UNKNOWN value and a direct write to ext-ERR<n>STATUS.OF that clears it to zero might indirectly set this bit to an UNKNOWN value.	
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERR<n>MISC0.CECR.	

Bits	Name	Description	Reset
[39]	OFR	Sticky overflow bit, repeat. Set to 1 when ERR<n>MISC0.CECR is incremented and wraps through zero. **0** Repeat counter has not overflowed. **1** Repeat counter has overflowed. A direct write that modifies this bit might indirectly set ext-ERR<n>STATUS.OF to an UNKNOWN value and a direct write to ext-ERR<n>STATUS.OF that clears it to zero might indirectly set this bit to an UNKNOWN value.	
[38:32]	CECR	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. Corrected errors are countable errors. It is **IMPLEMENTATION DEFINED** and might be UNPREDICTABLE whether Deferred and Uncorrected errors are countable errors. **Note:** For example, the other syndrome might include the set and way information for an error detected in a cache. This might be recorded in the **IMPLEMENTATION DEFINED** ERR<n>MISC<m> fields on a first Corrected error. ERR<n>MISC0.CECR is then incremented for each subsequent Corrected Error in the same set and way.	
[31:29]	WAY	The way that contained the error If the encoding of the way for the reported RAM requires fewer bits than the width of this field, the most significant bits of this field record the way, and the least significant bits are RES0.	
[28:19]	RES0	Reserved	0b0
[18:6]	INDX	The index that contained the error	
[5:4]	RES0	Reserved	0b0
[3:1]	LVL	Cache level **000** L1. **001** L2.	
[0]	InD	Instruction or Data cache **0** Data or unified cache. **1** Instruction cache.	

C.5.8 ERR1MISC0, Error Record Miscellaneous Register 0

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to identify the FRU in which the error was detected, and might contain enough information to locate the error within that FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

If the node that owns error record <n> implements architecturally-defined error counters (ERR<q>FR.CEC != 0b000), and error record <n> can record countable errors, then ERR<n>MISC0 implements the architecturally-defined error counter or counters.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure C-83: ext_err1misc0 bit assignments

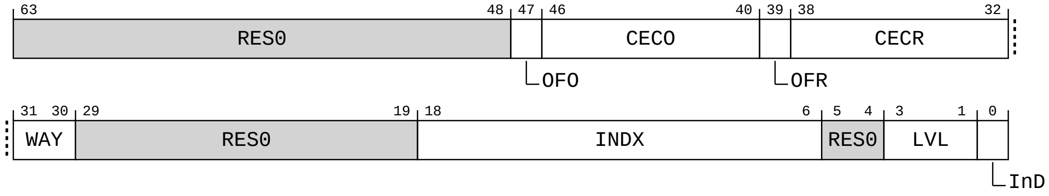


Table C-88: ERR1MISC0 bit descriptions

Bits	Name	Description	Reset
[63:48]	RES0	Reserved	0b0
[47]	OFO	Sticky overflow bit, other. Set to 1 when ERR<n>MISC0.CECO is incremented and wraps through zero. **0** Other counter has not overflowed. **1** Other counter has overflowed. A direct write that modifies this bit might indirectly set ext-ERR<n>STATUS.OF to an UNKNOWN value and a direct write to ext-ERR<n>STATUS.OF that clears it to zero might indirectly set this bit to an UNKNOWN value.	
[46:40]	CECO	Corrected error count, other. Incremented for each countable error that is not accounted for by incrementing ERR<n>MISC0.CECR.	

Bits	Name	Description	Reset

[39]	OFR	Sticky overflow bit, repeat. Set to 1 when ERR<n>MISC0.CECR is incremented and wraps through zero.
		`0`
		Repeat counter has not overflowed.
		`1`
		Repeat counter has overflowed.
		A direct write that modifies this bit might indirectly set ext-ERR<n>STATUS.OF to an UNKNOWN value and a direct write to ext-ERR<n>STATUS.OF that clears it to zero might indirectly set this bit to an UNKNOWN value.	

[38:32]	CECR	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. Corrected errors are countable errors. It is **IMPLEMENTATION DEFINED** and might be UNPREDICTABLE whether Deferred and Uncorrected errors are countable errors. **Note:** For example, the other syndrome might include the set and way information for an error detected in a cache. This might be recorded in the **IMPLEMENTATION DEFINED** ERR<n>MISC<m> fields on a first Corrected error. ERR<n>MISC0.CECR is then incremented for each subsequent Corrected Error in the same set and way.	
[31:30]	WAY	The way that contained the error	
[29:19]	RES0	Reserved	`0b0`
[18:6]	INDX	The index that contained the error	
[5:4]	RES0	Reserved	`0b0`

[3:1]	LVL	Cache level
		`000`
		L1.
		`001`
		L2.	

[0]	InD	Instruction or Data cache
		`0`
		Data or unified cache.
		`1`
		Instruction cache.	

C.5.9 ERR2PFGF, Pseudo-fault Generation Feature Register

Defines which common architecturally-defined fault generation features are implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure C-84: ext_err2pfgf bit assignments

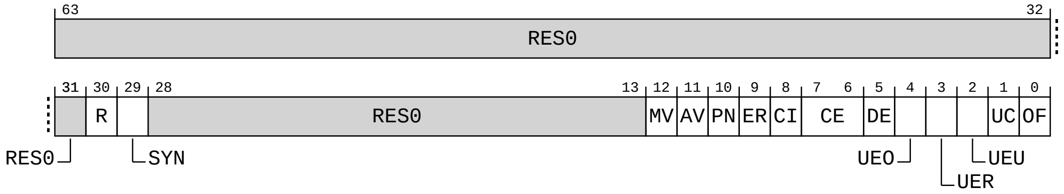


Table C-89: ERR2PFGF bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	0b0
[30]	R	Restartable. Support for Error Generation Counter restart mode. **1** Feature controllable.	
[29]	SYN	Syndrome. Fault syndrome injection. **0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.{IERR, SERR} to **IMPLEMENTATION DEFINED** values. ext-ERR<n>STATUS.{IERR, SERR} are UNKNOWN when ext-ERR<n>STATUS.V == 0b0.	
[28:13]	RES0	Reserved	0b0
[12]	MV	Miscellaneous syndrome. Additional syndrome injection. Defines whether software can control all or part of the syndrome recorded in the ERR<n>MISC<m> registers when an injected error is recorded. It is **IMPLEMENTATION DEFINED** which syndrome fields in ERR<n>MISC<m> this refers to, as some fields might always be recorded by an error. For example, a Corrected Error counter. **0** When an injected error is recorded, the node might record **IMPLEMENTATION DEFINED** additional syndrome in ERR<n>MISC<m>. If any syndrome is recorded in ERR<n>MISC<m>, then ext-ERR<n>STATUS.MV is set to 0b1.	
[11]	AV	Address syndrome. Address syndrome injection. **0** When an injected error is recorded, the node either sets ext-ERR<n>ADDR and ext-ERR<n>STATUS.AV for the access, or leaves these unchanged.	
[10]	PN	Poison flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.PN status flag. **0** When an injected error is recorded, it is **IMPLEMENTATION DEFINED** whether the node sets ext-ERR<n>STATUS.PN to 0b1.	

Bits	Name	Description	Reset
[9]	ER	Error Reported flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.ER status flag. **0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.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 ext-ERR<n>STATUS.CI status flag. **0** When an injected error is recorded, it is **IMPLEMENTATION DEFINED** whether the node sets ext-ERR<n>STATUS.CI to 0b1.	
[7:6]	CE	Corrected Error generation. Describes the types of Corrected Error that the fault generation feature of the node can generate. **01** The fault generation feature of the node allows generation of a non-specific Corrected Error, that is, a Corrected Error that is recorded as ext-ERR<n>STATUS.CE == 0b10.	
[5]	DE	Deferred Error generation. Describes whether the fault generation feature of the node can generate this type of error. **1** The fault generation feature of the node allows generation of this type of error.	
[4]	UEO	Latent or Restartable Error generation. Describes whether the fault generation feature of the node can generate this type of error. **0** The fault generation feature of the node cannot generate this type of error.	
[3]	UER	Signaled or Recoverable Error generation. Describes whether the fault generation feature of the node can generate this type of error. **0** The fault generation feature of the node cannot generate this type of error.	
[2]	UEU	Unrecoverable Error generation. Describes whether the fault generation feature of the node can generate this type of error. **0** The fault generation feature of the node cannot generate this type of error.	
[1]	UC	Uncontainable Error generation. Describes whether the fault generation feature of the node can generate this type of error. **1** 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 ext-ERR<n>STATUS.OF status flag. **0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.OF according to the architecture-defined rules for setting the OF bit.	

C.5.10 ERR1PFGF, Pseudo-fault Generation Feature Register

Defines which common architecturally-defined fault generation features are implemented.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure C-85: ext_err1pfgf bit assignments

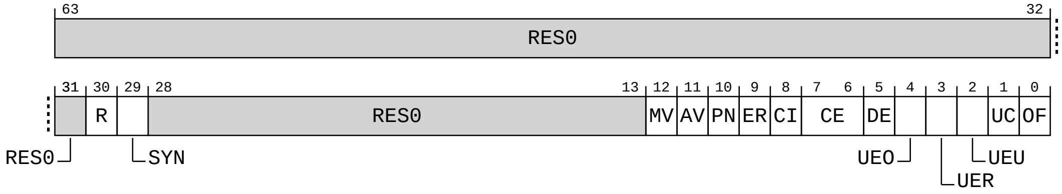


Table C-90: ERR1PFGF bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	`0b0`
[30]	R	Restartable. Support for Error Generation Counter restart mode. **1** Feature controllable.	
[29]	SYN	Syndrome. Fault syndrome injection. **0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.{IERR, SERR} to **IMPLEMENTATION DEFINED** values. ext-ERR<n>STATUS.{IERR, SERR} are UNKNOWN when ext-ERR<n>STATUS.V == 0b0.	
[28:13]	RES0	Reserved	`0b0`

Bits	Name	Description	Reset
[12]	MV	Miscellaneous syndrome. Additional syndrome injection. Defines whether software can control all or part of the syndrome recorded in the ERR<n>MISC<m> registers when an injected error is recorded. It is **IMPLEMENTATION DEFINED** which syndrome fields in ERR<n>MISC<m> this refers to, as some fields might always be recorded by an error. For example, a Corrected Error counter. **0** When an injected error is recorded, the node might record **IMPLEMENTATION DEFINED** additional syndrome in ERR<n>MISC<m>. If any syndrome is recorded in ERR<n>MISC<m>, then ext-ERR<n>STATUS.MV is set to 0b1.	
[11]	AV	Address syndrome. Address syndrome injection. **0** When an injected error is recorded, the node either sets ext-ERR<n>ADDR and ext-ERR<n>STATUS.AV for the access, or leaves these unchanged.	
[10]	PN	Poison flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.PN status flag. **0** When an injected error is recorded, it is **IMPLEMENTATION DEFINED** whether the node sets ext-ERR<n>STATUS.PN to 0b1.	
[9]	ER	Error Reported flag. Describes how the fault generation feature of the node sets the ext-ERR<n>STATUS.ER status flag. **0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.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 ext-ERR<n>STATUS.CI status flag. **0** When an injected error is recorded, it is **IMPLEMENTATION DEFINED** whether the node sets ext-ERR<n>STATUS.CI to 0b1.	
[7:6]	CE	Corrected Error generation. Describes the types of Corrected Error that the fault generation feature of the node can generate. **01** The fault generation feature of the node allows generation of a non-specific Corrected Error, that is, a Corrected Error that is recorded as ext-ERR<n>STATUS.CE == 0b10.	
[5]	DE	Deferred Error generation. Describes whether the fault generation feature of the node can generate this type of error. **1** The fault generation feature of the node allows generation of this type of error.	
[4]	UEO	Latent or Restartable Error generation. Describes whether the fault generation feature of the node can generate this type of error. **0** The fault generation feature of the node cannot generate this type of error.	
[3]	UER	Signaled or Recoverable Error generation. Describes whether the fault generation feature of the node can generate this type of error. **0** The fault generation feature of the node cannot generate this type of error.	

Bits	Name	Description	Reset
[2]	UEU	Unrecoverable Error generation. Describes whether the fault generation feature of the node can generate this type of error. **1** The fault generation feature of the node allows generation of this type of error.	
[1]	UC	Uncontainable Error generation. Describes whether the fault generation feature of the node can generate this type of error. **1** 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 ext-ERR<n>STATUS.OF status flag. **0** When an injected error is recorded, the node sets ext-ERR<n>STATUS.OF according to the architecture-defined rules for setting the OF bit.	

C.5.11 ERR2PFGCTL, Pseudo-fault Generation Control Register

Enables controlled fault generation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure C-86: ext_err2pfgctl bit assignments

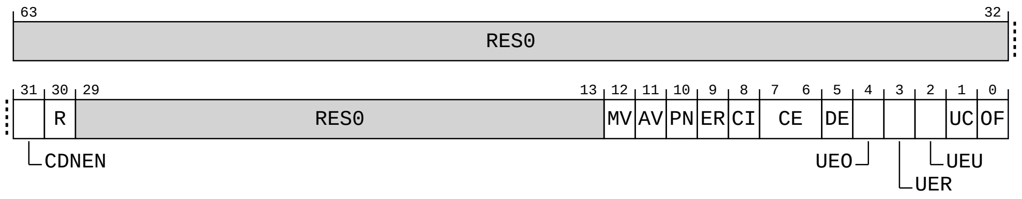


Table C-91: ERR2PFGCTL bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[31]	CDNEN	Countdown Enable. Controls transfers from the value that is held in the ext-ERR<n>PFGCDN into the Error Generation Counter and enables this counter. **0** The Error Generation Counter is disabled. **1** The Error Generation Counter is enabled. On a write of 0b1 to this bit, the Error Generation Counter is set to ext-ERR<n>PFGCDN.CDN.	`0b0`
[30]	R	Restart. Controls whether, upon reaching zero, the Error Generation Counter restarts from the ext-ERR<n>PFGCDN value or stops. **0** On reaching 0, the Error Generation Counter will stop. **1** On reaching 0, the Error Generation Counter is set to ext-ERR<n>PFGCDN.CDN. This bit is RES0 if the node does not support this control.	
[29:13]	RES0	Reserved	`0b0`
[12]	MV	Miscellaneous syndrome. The value that is written to ext-ERR<n>STATUS.MV when an injected error is recorded. **0** ext-ERR<n>STATUS.MV is set to 0b0 when an injected error is recorded. **1** ext-ERR<n>STATUS.MV is set to 0b1 when an injected error is recorded. This bit reads-as-one if the node always records some syndrome in ERR<n>MISC<m>, setting ext-ERR<n>STATUS.MV to 1, when an injected error is recorded. This bit is RES0 if the node does not support this control.	
[11]	AV	Address syndrome. The value that is written to ext-ERR<n>STATUS.AV when an injected error is recorded. **0** ext-ERR<n>STATUS.AV is set to 0b0 when an injected error is recorded. **1** ext-ERR<n>STATUS.AV is set to 0b1 when an injected error is recorded. This bit reads-as-one if the node always sets ext-ERR<n>STATUS.AV to 0b1 when an injected error is recorded. This bit is RES0 if the node does not support this control.	
[10]	PN	Poison flag. The value that is written to ext-ERR<n>STATUS.PN when an injected error is recorded. **0** ext-ERR<n>STATUS.PN is set to 0b0 when an injected error is recorded. **1** ext-ERR<n>STATUS.PN is set to 0b1 when an injected error is recorded. This bit is RES0 if the node does not support this control.	

Bits	Name	Description	Reset
[9]	ER	Error Reported flag. The value that is written to ext-ERR<n>STATUS.ER when an injected error is recorded. **0** ext-ERR<n>STATUS.ER is set to 0b0 when an injected error is recorded. **1** ext-ERR<n>STATUS.ER is set to 0b1 when an injected error is recorded. This bit is RES0 if the node does not support this control.	
[8]	CI	Critical Error flag. The value that is written to ext-ERR<n>STATUS.CI when an injected error is recorded. **0** ext-ERR<n>STATUS.CI is set to 0b0 when an injected error is recorded. **1** ext-ERR<n>STATUS.CI is set to 0b1 when an injected error is recorded. This bit is RES0 if the node does not support this control.	
[7:6]	CE	Corrected Error generation enable. Controls the type of Corrected Error condition that might be generated. **00** No error of this type will be generated. **01** A non-specific Corrected Error, that is, a Corrected Error that is recorded as ext-ERR<n>STATUS.CE == 0b10, might be generated when the Error Generation Counter decrements to zero. **10** A transient Corrected Error, that is, a Corrected Error that is recorded as ext-ERR<n>STATUS.CE == 0b01, might be generated when the Error Generation Counter decrements to zero. **11** A persistent Corrected Error, that is, a Corrected Error that is recorded as ext-ERR<n>STATUS.CE == 0b11, might be generated when the Error Generation Counter decrements to zero. The set of permitted values for this field is defined by ext-ERR<n>PFGF.CE. This field is RES0 if the node does not support this control.	
[5]	DE	Deferred Error generation enable. Controls whether this type of error condition might be generated. It is **IMPLEMENTATION DEFINED** whether the error is generated if the data is not consumed. **0** No error of this type will be generated. **1** An error of this type might be generated when the Error Generation Counter decrements to zero. This bit is RES0 if the node does not support this control.	
[4]	UEO	Latent or Restartable Error generation enable. Controls whether this type of error condition might be generated. It is **IMPLEMENTATION DEFINED** whether the error is generated if the data is not consumed. **0** No error of this type will be generated. **1** An error of this type might be generated when the Error Generation Counter decrements to zero. This bit is RES0 if the node does not support this control.	

Bits	Name	Description	Reset
[3]	UER	Signaled or Recoverable Error generation enable. Controls whether this type of error condition might be generated. It is **IMPLEMENTATION DEFINED** whether the error is generated if the data is not consumed. **0** No error of this type will be generated. **1** An error of this type might be generated when the Error Generation Counter decrements to zero. This bit is RES0 if the node does not support this control.	
[2]	UEU	Unrecoverable Error generation enable. Controls whether this type of error condition might be generated. It is **IMPLEMENTATION DEFINED** whether the error is generated if the data is not consumed. **0** No error of this type will be generated. **1** An error of this type might be generated when the Error Generation Counter decrements to zero. This bit is RES0 if the node does not support this control.	
[1]	UC	Uncontainable Error generation enable. Controls whether this type of error condition might be generated. It is **IMPLEMENTATION DEFINED** whether the error is generated if the data is not consumed. **0** No error of this type will be generated. **1** An error of this type might be generated when the Error Generation Counter decrements to zero. This bit is RES0 if the node does not support this control.	
[0]	OF	Overflow flag. The value that is written to ext-ERR<n>STATUS.OF when an injected error is recorded. **0** ext-ERR<n>STATUS.OF is set to 0b0 when an injected error is recorded. **1** ext-ERR<n>STATUS.OF is set to 0b1 when an injected error is recorded. This bit is RES0 if the node does not support this control.	

C.5.12 ERR1PFGCTL, Pseudo-fault Generation Control Register

Enables controlled fault generation.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure C-87: ext_err1pfgctl bit assignments

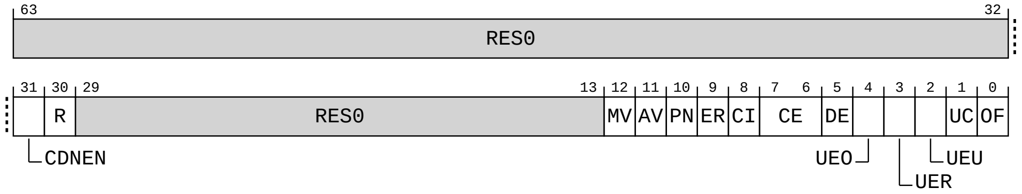


Table C-92: ERR1PFGCTL bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	`0b0`
[31]	CDNEN	Countdown Enable. Controls transfers from the value that is held in the ext-ERR<n>PFGCDN into the Error Generation Counter and enables this counter. **0** The Error Generation Counter is disabled. **1** The Error Generation Counter is enabled. On a write of 0b1 to this bit, the Error Generation Counter is set to ext-ERR<n>PFGCDN.CDN.	`0b0`
[30]	R	Restart. Controls whether, upon reaching zero, the Error Generation Counter restarts from the ext-ERR<n>PFGCDN value or stops. **0** On reaching 0, the Error Generation Counter will stop. **1** On reaching 0, the Error Generation Counter is set to ext-ERR<n>PFGCDN.CDN. This bit is RES0 if the node does not support this control.	
[29:13]	RES0	Reserved	`0b0`
[12]	MV	Miscellaneous syndrome. The value that is written to ext-ERR<n>STATUS.MV when an injected error is recorded. **0** ext-ERR<n>STATUS.MV is set to 0b0 when an injected error is recorded. **1** ext-ERR<n>STATUS.MV is set to 0b1 when an injected error is recorded. This bit reads-as-one if the node always records some syndrome in ERR<n>MISC<m>, setting ext-ERR<n>STATUS.MV to 1, when an injected error is recorded. This bit is RES0 if the node does not support this control.	

Bits	Name	Description	Reset

[11]	AV	Address syndrome. The value that is written to ext-ERR<n>STATUS.AV when an injected error is recorded.
		`0`
		ext-ERR<n>STATUS.AV is set to 0b0 when an injected error is recorded.
		`1`
		ext-ERR<n>STATUS.AV is set to 0b1 when an injected error is recorded.
		This bit reads-as-one if the node always sets ext-ERR<n>STATUS.AV to 0b1 when an injected error is recorded. This bit is RES0 if the node does not support this control.	

[10]	PN	Poison flag. The value that is written to ext-ERR<n>STATUS.PN when an injected error is recorded.
		`0`
		ext-ERR<n>STATUS.PN is set to 0b0 when an injected error is recorded.
		`1`
		ext-ERR<n>STATUS.PN is set to 0b1 when an injected error is recorded.
		This bit is RES0 if the node does not support this control.	

[9]	ER	Error Reported flag. The value that is written to ext-ERR<n>STATUS.ER when an injected error is recorded.
		`0`
		ext-ERR<n>STATUS.ER is set to 0b0 when an injected error is recorded.
		`1`
		ext-ERR<n>STATUS.ER is set to 0b1 when an injected error is recorded.
		This bit is RES0 if the node does not support this control.	

[8]	CI	Critical Error flag. The value that is written to ext-ERR<n>STATUS.CI when an injected error is recorded.
		`0`
		ext-ERR<n>STATUS.CI is set to 0b0 when an injected error is recorded.
		`1`
		ext-ERR<n>STATUS.CI is set to 0b1 when an injected error is recorded.
		This bit is RES0 if the node does not support this control.	

[7:6]	CE	Corrected Error generation enable. Controls the type of Corrected Error condition that might be generated.
		`00`
		No error of this type will be generated.
		`01`
		A non-specific Corrected Error, that is, a Corrected Error that is recorded as ext-ERR<n>STATUS.CE == 0b10, might be generated when the Error Generation Counter decrements to zero.
		`10`
		A transient Corrected Error, that is, a Corrected Error that is recorded as ext-ERR<n>STATUS.CE == 0b01, might be generated when the Error Generation Counter decrements to zero.
		`11`
		A persistent Corrected Error, that is, a Corrected Error that is recorded as ext-ERR<n>STATUS.CE == 0b11, might be generated when the Error Generation Counter decrements to zero.
		The set of permitted values for this field is defined by ext-ERR<n>PFGF.CE.
		This field is RES0 if the node does not support this control.	

Bits	Name	Description	Reset
[5]	DE	Deferred Error generation enable. Controls whether this type of error condition might be generated. It is **IMPLEMENTATION DEFINED** whether the error is generated if the data is not consumed. **0** No error of this type will be generated. **1** An error of this type might be generated when the Error Generation Counter decrements to zero. This bit is RES0 if the node does not support this control.	
[4]	UEO	Latent or Restartable Error generation enable. Controls whether this type of error condition might be generated. It is **IMPLEMENTATION DEFINED** whether the error is generated if the data is not consumed. **0** No error of this type will be generated. **1** An error of this type might be generated when the Error Generation Counter decrements to zero. This bit is RES0 if the node does not support this control.	
[3]	UER	Signaled or Recoverable Error generation enable. Controls whether this type of error condition might be generated. It is **IMPLEMENTATION DEFINED** whether the error is generated if the data is not consumed. **0** No error of this type will be generated. **1** An error of this type might be generated when the Error Generation Counter decrements to zero. This bit is RES0 if the node does not support this control.	
[2]	UEU	Unrecoverable Error generation enable. Controls whether this type of error condition might be generated. It is **IMPLEMENTATION DEFINED** whether the error is generated if the data is not consumed. **0** No error of this type will be generated. **1** An error of this type might be generated when the Error Generation Counter decrements to zero. This bit is RES0 if the node does not support this control.	
[1]	UC	Uncontainable Error generation enable. Controls whether this type of error condition might be generated. It is **IMPLEMENTATION DEFINED** whether the error is generated if the data is not consumed. **0** No error of this type will be generated. **1** An error of this type might be generated when the Error Generation Counter decrements to zero. This bit is RES0 if the node does not support this control.	

Bits	Name	Description	Reset
[0]	OF	Overflow flag. The value that is written to ext-ERR<n>STATUS.OF when an injected error is recorded. **0** ext-ERR<n>STATUS.OF is set to 0b0 when an injected error is recorded. **1** ext-ERR<n>STATUS.OF is set to 0b1 when an injected error is recorded. This bit is RES0 if the node does not support this control.	

C.5.13 ERR2STATUS, Error Record Primary Status Register

Contains status information for the error record, including:

- Whether any error has been detected (valid).
- Whether any detected error was not corrected, and returned to a master.
- Whether any detected error was not corrected and deferred.
- Whether an error record has been discarded because additional errors have been detected before the first error was handled by software (overflow).
- Whether any error has been reported.
- Whether the other error record registers contain valid information.
- Whether the error was reported because poison data was detected or because a corrupt value was detected by an error detection code.
- A primary error code.
- An **IMPLEMENTATION DEFINED** extended error code.

Within this register:

- The {AV, V, MV} bits are valid bits that define whether the error record registers are valid.
- The {UE, OF, CE, DE, UET} bits encode the types of error or errors recorded.
- The {CI, ER, PN, IERR, SERR} fields are syndrome fields.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure C-88: ext_err2status bit assignments

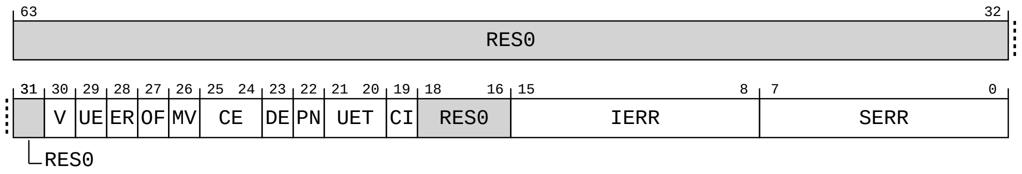


Table C-93: ERR2STATUS bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	0b0
[30]	V	Status Register Valid. **0** ERR<n>STATUS not valid. **1** ERR<n>STATUS valid. At least one error has been recorded. This bit is read/write-one-to-clear.	0b0
[29]	UE	Uncorrected Error. **0** No errors have been detected, or all detected errors have been either corrected or deferred. **1** At least one detected error was not corrected and not deferred. When clearing ERR<n>STATUS.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if ERR<n>STATUS.V == 0b0. This bit is read/write-one-to-clear.	

Bits	Name	Description	Reset
[28]	ER	Error Reported. **0** No in-band error (External Abort) reported. **1** An External Abort was signaled by the node to the master making the access or other transaction. This can be because any of the following are true: • The applicable one of the `ERR<q>CTLR`.{WUE,RUE,UE} bits is implemented and was set to 0b1 when an Uncorrected error was detected. • The applicable one of the `ERR<q>CTLR`.{WUE,RUE,UE} bits is not implemented and the node always reports errors. It is **IMPLEMENTATION DEFINED** whether this bit can be set to 0b1 by a Deferred error. When clearing ERR<n>STATUS.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if any of the following are true: • ERR<n>STATUS.V == 0b0. • ERR<n>STATUS.UE == 0b0 and this bit is never set to 0b1 by a Deferred error. • ERR<n>STATUS.{UE,DE} == {0,0} and this bit can be set to 0b1 by a Deferred error. This bit is read/write-one-to-clear. **Note:** An External Abort signaled by the node might be masked and not generate any exception.	

Bits	Name	Description	Reset
[27]	OF	Overflow. Indicates that multiple errors have been detected. This bit is set to 0b1 when one of the following occurs: • A Corrected error counter is implemented, an error is counted, and the counter overflows. • ERR<n>STATUS.V was previously set to 0b1, a Corrected error counter is not implemented, and a Corrected error is recorded. • ERR<n>STATUS.V was previously set to 0b1, and a type of error other than a Corrected error is recorded. Otherwise, this bit is unchanged when an error is recorded. If a Corrected error counter is implemented: ◦ A direct write that modifies the counter overflow flag indirectly might set this bit to an UNKNOWN value. ◦ A direct write to this bit that clears this bit to zero might indirectly set the counter overflow flag to an UNKNOWN value. **0** Since this bit was last cleared to zero, no error syndrome has been discarded and, if a Corrected error counter is implemented, it has not overflowed. **1** Since this bit was last cleared to zero, at least one error syndrome has been discarded or, if a Corrected error counter is implemented, it might have overflowed. When clearing ERR<n>STATUS.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if ERR<n>STATUS.V == 0b0. This bit is read/write-one-to-clear.	
[26]	MV	Miscellaneous Registers Valid. **0** ERR<n>MISC<m> not valid. **1** The **IMPLEMENTATION DEFINED** contents of the ERR<n>MISC<m> registers contains additional information for an error recorded by this record. This bit is read/write-one-to-clear. **Note:** If the ERR<n>MISC<m> registers can contain additional information for a previously recorded error, then the contents must be self-describing to software or a user. For example, certain fields might relate only to Corrected errors, and other fields only to the most recent error that was not discarded.	`0b0`

Bits	Name	Description	Reset
[25:24]	CE	Corrected Error. **`00`** No errors were corrected. **`01`** At least one transient error was corrected. **`10`** At least one error was corrected. **`11`** At least one persistent error was corrected. The mechanism by which a node detects whether a correctable error is transient or persistent is **IMPLEMENTATION DEFINED**. If no such mechanism is implemented, then the node sets this field to 0b10 when an error is corrected. When clearing ERR<n>STATUS.V to 0b0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero. This field is not valid and reads UNKNOWN if ERR<n>STATUS.V == 0b0. 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.	
[23]	DE	Deferred Error. **`0`** No errors were deferred. **`1`** At least one error was not corrected and deferred. Support for deferring errors is **IMPLEMENTATION DEFINED**. When clearing ERR<n>STATUS.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if ERR<n>STATUS.V == 0b0. This bit is read/write-one-to-clear.	

Bits	Name	Description	Reset
[22]	PN	Poison. **0** Uncorrected error or Deferred error recorded because a corrupt value was detected, for example, by an error detection code (EDC). **Note:** If a producer node detects a corrupt value and defers the error by producing a poison value, then this bit is set to 0b0 at the producer node. **1** Uncorrected error or Deferred error recorded because a poison value was detected. **Note:** This might only be an indication of poison, because, in some EDC schemes, a poison value is encoded as an unlikely form of corrupt data, meaning it is possible to mistake a corrupt value as a poison value. It is **IMPLEMENTATION DEFINED** whether a node can distinguish a poison value from a corrupt value. When clearing ERR<n>STATUS.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if any of the following are true: • ERR<n>STATUS.V == 0b0. • ERR<n>STATUS.{DE,UE} == {0,0}. This bit is read/write-one-to-clear.	
[21:20]	UET	Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error. **00** Uncorrected error, Uncontainable error (UC). **01** Uncorrected error, Unrecoverable error (UEU). **10** Uncorrected error, Latent or Restartable error (UEO). **11** Uncorrected error, Signaled or Recoverable error (UER). When clearing ERR<n>STATUS.V to 0b0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero. This field is not valid and reads UNKNOWN if any of the following are true: • ERR<n>STATUS.V == 0b0. • ERR<n>STATUS.UE == 0b0. 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. **Note:** Software might use the information in the error record registers to determine what recovery is necessary.	
[19]	CI	Critical Error. Indicates whether a critical error condition has been recorded. **0** No critical error condition.	

Bits	Name	Description	Reset
[18:16]	RES0	Reserved	`0b0`
[15:8]	IERR	**IMPLEMENTATION DEFINED** error code. Used with any primary error code ERR<n>STATUS.SERR value. Further **IMPLEMENTATION DEFINED** information can be placed in the ERR<n>MISC<m> registers. The implemented set of valid values that this field can take is **IMPLEMENTATION DEFINED**. If any value not in this set is written to this register, then the value read back from this field is UNKNOWN. **Note:** This means that one or more bits of this field might be implemented as fixed read-as-zero or read-as-one values. This field is not valid and reads UNKNOWN if all of the following are true: • Any of the following are true: ◦ The RAS Common Fault Injection Model Extension is implemented by the node that owns this error record and `ERR<q>PFGF`.SYN == 0b0. ◦ The RAS Common Fault Injection Model Extension is not implemented by the node that owns this error record. • ERR<n>STATUS.V == 0b0.	
[7:0]	SERR	Architecturally-defined primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry. **`00000110`** Data value from associative memory. For example, ECC error on cache data. **`00000111`** Address/control value from associative memory. For example, ECC error on cache tag. **`00001000`** Data value from a TLB. For example, ECC error on TLB data. **`00001100`** Data value from (non-associative) external memory. For example, ECC error in SDRAM. **`00010010`** Error response from slave. For example, error response from cache write-back. **`00010101`** Deferred error from slave not supported at master. For example, poisoned data received from a slave by a master that cannot defer the error further.	

C.5.14 ERR1STATUS, Error Record Primary Status Register

Contains status information for the error record, including:

- Whether any error has been detected (valid).
- Whether any detected error was not corrected, and returned to a master.
- Whether any detected error was not corrected and deferred.
- Whether an error record has been discarded because additional errors have been detected before the first error was handled by software (overflow).
- Whether any error has been reported.
- Whether the other error record registers contain valid information.

- Whether the error was reported because poison data was detected or because a corrupt value was detected by an error detection code.
- A primary error code.
- An **IMPLEMENTATION DEFINED** extended error code.

Within this register:

- The {AV, V, MV} bits are valid bits that define whether the error record registers are valid.
- The {UE, OF, CE, DE, UET} bits encode the types of error or errors recorded.
- The {CI, ER, PN, IERR, SERR} fields are syndrome fields.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

See individual bit resets.

Bit descriptions

Figure C-89: ext_err1status bit assignments

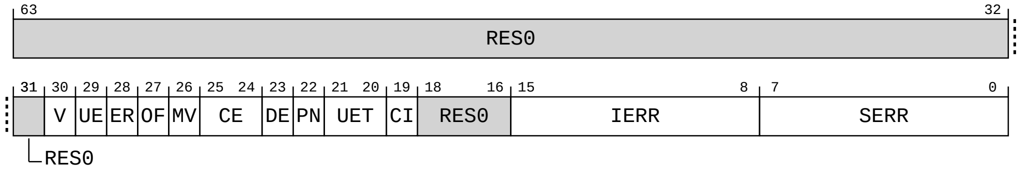


Table C-94: ERR1STATUS bit descriptions

Bits	Name	Description	Reset
[63:31]	RES0	Reserved	`0b0`
[30]	V	Status Register Valid. **0** ERR<n>STATUS not valid. **1** ERR<n>STATUS valid. At least one error has been recorded. This bit is read/write-one-to-clear.	`0b0`

Bits	Name	Description	Reset
[29]	UE	Uncorrected Error. **0** No errors have been detected, or all detected errors have been either corrected or deferred. **1** At least one detected error was not corrected and not deferred. When clearing ERR<n>STATUS.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if ERR<n>STATUS.V == 0b0. This bit is read/write-one-to-clear.	
[28]	ER	Error Reported. **0** No in-band error (External Abort) reported. **1** An External Abort was signaled by the node to the master making the access or other transaction. This can be because any of the following are true: • The applicable one of the `ERR<q>CTLR`.{WUE,RUE,UE} bits is implemented and was set to 0b1 when an Uncorrected error was detected. • The applicable one of the `ERR<q>CTLR`.{WUE,RUE,UE} bits is not implemented and the node always reports errors. It is **IMPLEMENTATION DEFINED** whether this bit can be set to 0b1 by a Deferred error. When clearing ERR<n>STATUS.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if any of the following are true: • ERR<n>STATUS.V == 0b0. • ERR<n>STATUS.UE == 0b0 and this bit is never set to 0b1 by a Deferred error. • ERR<n>STATUS.{UE,DE} == {0,0} and this bit can be set to 0b1 by a Deferred error. This bit is read/write-one-to-clear. **Note:** An External Abort signaled by the node might be masked and not generate any exception.	

Bits	Name	Description	Reset
[27]	OF	Overflow. Indicates that multiple errors have been detected. This bit is set to 0b1 when one of the following occurs: • A Corrected error counter is implemented, an error is counted, and the counter overflows. • ERR<n>STATUS.V was previously set to 0b1, a Corrected error counter is not implemented, and a Corrected error is recorded. • ERR<n>STATUS.V was previously set to 0b1, and a type of error other than a Corrected error is recorded. Otherwise, this bit is unchanged when an error is recorded. If a Corrected error counter is implemented: ◦ A direct write that modifies the counter overflow flag indirectly might set this bit to an UNKNOWN value. ◦ A direct write to this bit that clears this bit to zero might indirectly set the counter overflow flag to an UNKNOWN value. **0** Since this bit was last cleared to zero, no error syndrome has been discarded and, if a Corrected error counter is implemented, it has not overflowed. **1** Since this bit was last cleared to zero, at least one error syndrome has been discarded or, if a Corrected error counter is implemented, it might have overflowed. When clearing ERR<n>STATUS.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if ERR<n>STATUS.V == 0b0. This bit is read/write-one-to-clear.	
[26]	MV	Miscellaneous Registers Valid. **0** ERR<n>MISC<m> not valid. **1** The **IMPLEMENTATION DEFINED** contents of the ERR<n>MISC<m> registers contains additional information for an error recorded by this record. This bit is read/write-one-to-clear. **Note:** If the ERR<n>MISC<m> registers can contain additional information for a previously recorded error, then the contents must be self-describing to software or a user. For example, certain fields might relate only to Corrected errors, and other fields only to the most recent error that was not discarded.	`0b0`

Bits	Name	Description	Reset
[25:24]	CE	Corrected Error. **00** No errors were corrected. **01** At least one transient error was corrected. **10** At least one error was corrected. **11** At least one persistent error was corrected. The mechanism by which a node detects whether a correctable error is transient or persistent is **IMPLEMENTATION DEFINED**. If no such mechanism is implemented, then the node sets this field to 0b10 when an error is corrected. When clearing ERR<n>STATUS.V to 0b0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero. This field is not valid and reads UNKNOWN if ERR<n>STATUS.V == 0b0. 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.	
[23]	DE	Deferred Error. **0** No errors were deferred. **1** At least one error was not corrected and deferred. Support for deferring errors is **IMPLEMENTATION DEFINED**. When clearing ERR<n>STATUS.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if ERR<n>STATUS.V == 0b0. This bit is read/write-one-to-clear.	

Bits	Name	Description	Reset
[22]	PN	Poison. **0** Uncorrected error or Deferred error recorded because a corrupt value was detected, for example, by an error detection code (EDC). **Note:** If a producer node detects a corrupt value and defers the error by producing a poison value, then this bit is set to 0b0 at the producer node. **1** Uncorrected error or Deferred error recorded because a poison value was detected. **Note:** This might only be an indication of poison, because, in some EDC schemes, a poison value is encoded as an unlikely form of corrupt data, meaning it is possible to mistake a corrupt value as a poison value. It is **IMPLEMENTATION DEFINED** whether a node can distinguish a poison value from a corrupt value. When clearing ERR<n>STATUS.V to 0b0, if this bit is nonzero, then Arm recommends that software write 0b1 to this bit to clear this bit to zero. This bit is not valid and reads UNKNOWN if any of the following are true: • ERR<n>STATUS.V == 0b0. • ERR<n>STATUS.{DE,UE} == {0,0}. This bit is read/write-one-to-clear.	
[21:20]	UET	Uncorrected Error Type. Describes the state of the component after detecting or consuming an Uncorrected error. **00** Uncorrected error, Uncontainable error (UC). **01** Uncorrected error, Unrecoverable error (UEU). **10** Uncorrected error, Latent or Restartable error (UEO). **11** Uncorrected error, Signaled or Recoverable error (UER). When clearing ERR<n>STATUS.V to 0b0, if this field is nonzero, then Arm recommends that software write ones to this field to clear this field to zero. This field is not valid and reads UNKNOWN if any of the following are true: • ERR<n>STATUS.V == 0b0. • ERR<n>STATUS.UE == 0b0. 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. **Note:** Software might use the information in the error record registers to determine what recovery is necessary.	
[19]	CI	Critical Error. Indicates whether a critical error condition has been recorded. **0** No critical error condition.	

Bits	Name	Description	Reset
[18:16]	RES0	Reserved	`0b0`
[15:8]	IERR	**IMPLEMENTATION DEFINED** error code. Used with any primary error code ERR<n>STATUS.SERR value. Further **IMPLEMENTATION DEFINED** information can be placed in the ERR<n>MISC<m> registers. The implemented set of valid values that this field can take is **IMPLEMENTATION DEFINED**. If any value not in this set is written to this register, then the value read back from this field is UNKNOWN. **Note:** This means that one or more bits of this field might be implemented as fixed read-as-zero or read-as-one values. This field is not valid and reads UNKNOWN if all of the following are true: • Any of the following are true: ◦ The RAS Common Fault Injection Model Extension is implemented by the node that owns this error record and `ERR<q>PFGF`.SYN == 0b0. ◦ The RAS Common Fault Injection Model Extension is not implemented by the node that owns this error record. • ERR<n>STATUS.V == 0b0.	
[7:0]	SERR	Architecturally-defined primary error code. The primary error code might be used by a fault handling agent to triage an error without requiring device-specific code. For example, to count and threshold corrected errors in software, or generate a short log entry. **`00000110`** Data value from associative memory. For example, ECC error on cache data. **`00000111`** Address/control value from associative memory. For example, ECC error on cache tag. **`00001000`** Data value from a TLB. For example, ECC error on TLB data. **`00001100`** Data value from (non-associative) external memory. For example, ECC error in SDRAM. **`00010010`** Error response from slave. For example, error response from cache write-back. **`00010101`** Deferred error from slave not supported at master. For example, poisoned data received from a slave by a master that cannot defer the error further.	

C.5.15 ERR2MISC3, Error Record Miscellaneous Register 3

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to identify the FRU in which the error was detected, and might contain enough information to locate the error within that FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

If the node that owns error record n supports the RAS Timestamp Extension (`ERR<q>FR`.TS != 0b00), then ERR<n>MISC3 contains the timestamp value for error record n when the error was detected. Otherwise the contents of ERR<n>MISC3 are **IMPLEMENTATION DEFINED**.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

`0x0`

Bit descriptions

Figure C-90: ext_err2misc3 bit assignments

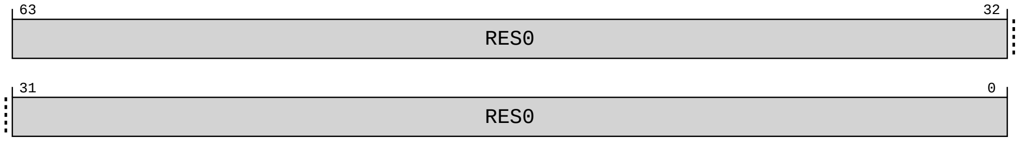


Table C-95: ERR2MISC3 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	`0b0`

C.5.16 ERR1MISC3, Error Record Miscellaneous Register 3

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to identify the FRU in which the error was detected, and might contain enough information to locate the error within that FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

If the node that owns error record n supports the RAS Timestamp Extension (`ERR<q>FR`.TS != 0b00), then ERR<n>MISC3 contains the timestamp value for error record n when the error was detected. Otherwise the contents of ERR<n>MISC3 are **IMPLEMENTATION DEFINED**.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

`0x0`

Bit descriptions

Figure C-91: ext_err1misc3 bit assignments

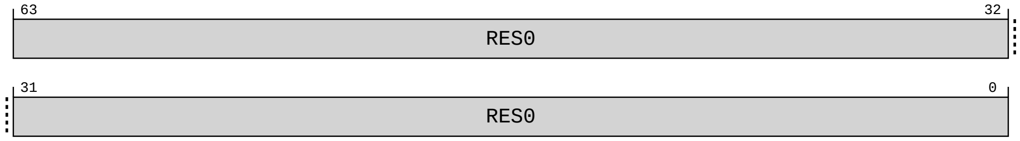


Table C-96: ERR1MISC3 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	`0b0`

C.5.17 ERR2MISC2, Error Record Miscellaneous Register 2

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to identify the FRU in which the error was detected, and might contain enough information to locate the error within that FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

`0x0`

Bit descriptions

Figure C-92: ext_err2misc2 bit assignments

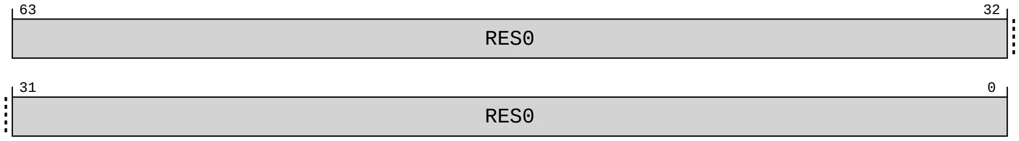


Table C-97: ERR2MISC2 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	`0b0`

C.5.18 ERR1MISC2, Error Record Miscellaneous Register 2

IMPLEMENTATION DEFINED error syndrome register. The miscellaneous syndrome registers might contain:

- Information to identify the FRU in which the error was detected, and might contain enough information to locate the error within that FRU.
- A Corrected error counter or counters.
- Other state information not present in the corresponding status and address registers.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

ras

Reset value

`0x0`

Bit descriptions

Figure C-93: ext_err1misc2 bit assignments

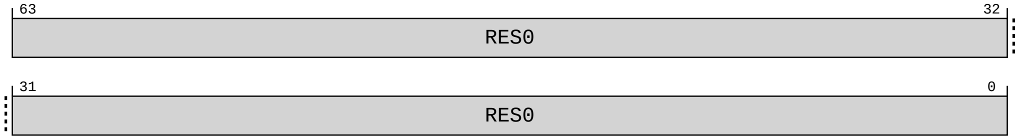


Table C-98: ERR1MISC2 bit descriptions

Bits	Name	Description	Reset
[63:0]	RES0	Reserved	0b0

C.6 ETE register summary

The summary table provides an overview of memory-mapped *Embedded Trace Extension* (ETE) registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table C-99: ETE register summary

Offset	Name	Reset	Width	Description
0x18	TRCAUXCTLR	0x0	32-bit	Auxiliary Control Register
0x180	TRCIDR8	See individual bit resets	32-bit	ID Register 8
0x184	TRCIDR9	0x0	32-bit	ID Register 9
0x188	TRCIDR10	0x0	32-bit	ID Register 10
0x18C	TRCIDR11	0x0	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	0x0	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

Offset	Name	Reset	Width	Description
`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

C.6.1 TRCAUXCTLR, Auxiliary 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 C-94: ext_trcauxctlr bit assignments

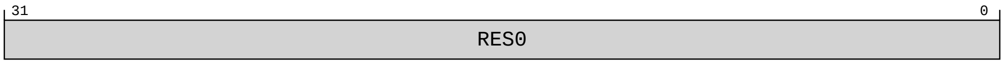


Table C-100: TRCAUXCTLR bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.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 C-95: ext_trcidr8 bit assignments

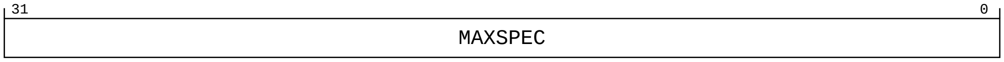


Table C-101: TRCIDR8 bit descriptions

Bits	Name	Description	Reset
[31:0]	MAXSPEC	Indicates the maximum speculation depth of the instruction trace element stream. This is the maximum number of P0 elements in the trace element stream that can be speculative at any time. `00000000000000000000000000000000`	

C.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

`0x0`

Bit descriptions

Figure C-96: ext_trcidr9 bit assignments

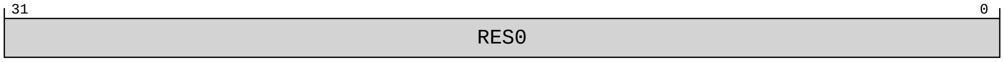


Table C-102: TRCIDR9 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.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

ETE

Register offset

0x188

Reset value

`0x0`

Bit descriptions

Figure C-97: ext_trcidr10 bit assignments

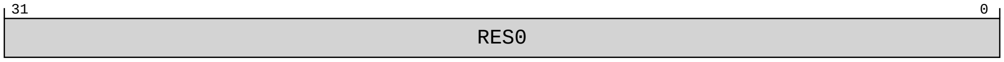


Table C-103: TRCIDR10 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.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

`0x0`

Bit descriptions

Figure C-98: ext_trcidr11 bit assignments

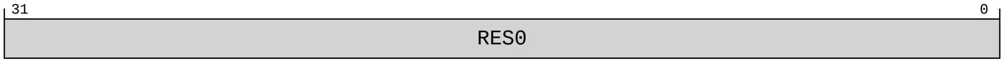


Table C-104: TRCIDR11 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.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 C-99: ext_trcidr12 bit assignments

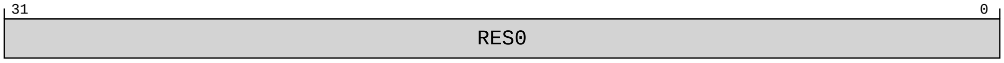


Table C-105: TRCIDR12 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.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 C-100: ext_trcidr13 bit assignments

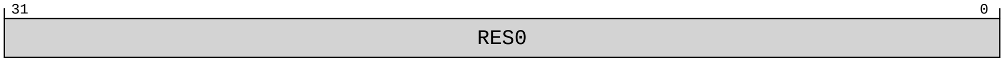


Table C-106: TRCIDR13 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

C.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 C-101: ext_trcimspec0 bit assignments

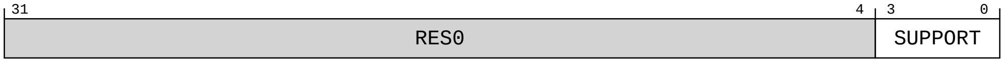


Table C-107: TRCIMSPEC0 bit descriptions

Bits	Name	Description	Reset
[31:4]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[3:0]	SUPPORT	Indicates whether the implementation supports **IMPLEMENTATION DEFINED** features. `0000` No **IMPLEMENTATION DEFINED** features are supported.	

C.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 C-102: ext_trcidr0 bit assignments

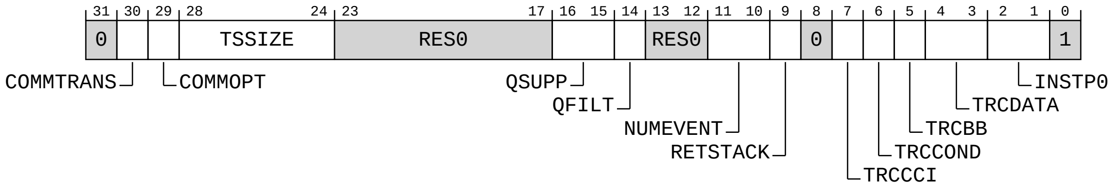


Table C-108: TRCIDR0 bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	`0b0`
[30]	COMMTRANS	Transaction Start element behavior. `0` Transaction Start elements are P0 elements.	
[29]	COMMOPT	Indicates the contents and encodings of Cycle count packets. `1` Commit mode 1.	

Bits	Name	Description	Reset
[28:24]	TSSIZE	Indicates that the trace unit implements Global timestamping and the size of the timestamp value. **01000** Global timestamping implemented with a 64-bit timestamp value.	
[23:17]	RES0	Reserved	0b0
[16:15]	QSUPP	Indicates that the trace unit implements Q element support. **00** Q element support is not implemented.	
[14]	QFILT	Indicates if the trace unit implements Q element filtering. **0** Q element filtering is not implemented.	
[13:12]	RES0	Reserved	0b0
[11:10]	NUMEVENT	Indicates the number of ETEEvents implemented. **11** The trace unit supports 4 ETEEvents.	
[9]	RETSTACK	Indicates if the trace unit supports the return stack. **1** Return stack implemented.	
[8]	RES0	Reserved	0b0
[7]	TRCCCI	Indicates if the trace unit implements cycle counting. **1** 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. **0** Conditional instruction tracing not implemented.	
[5]	TRCBB	Indicates if the trace unit implements branch broadcasting. **1** 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. **00** Tracing of data addresses and data values is not implemented.	
[2:1]	INSTP0	Indicates if load and store instructions are P0 instructions. Load and store instructions as P0 instructions is not implemented in ETE and this field is reserved for other trace architectures. **00** Load and store instructions are not P0 instructions.	
[0]	RES1	Reserved	0b1

C.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 C-103: ext_trcidr1 bit assignments

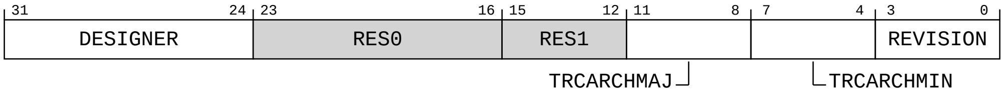


Table C-109: TRCIDR1 bit descriptions

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. **01000001** Arm Limited	
[23:16]	RES0	Reserved	0b0
[15:12]	RES1	Reserved	0b1
[11:8]	TRCARCHMAJ	Major architecture version. **1111** If both TRCARCHMAJ and TRCARCHMIN == 0xF then refer to ext-TRCDEVARCH.	
[7:4]	TRCARCHMIN	Minor architecture version. **1111** 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. **0000** Revision 0	

C.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 C-104: ext_trcidr2 bit assignments

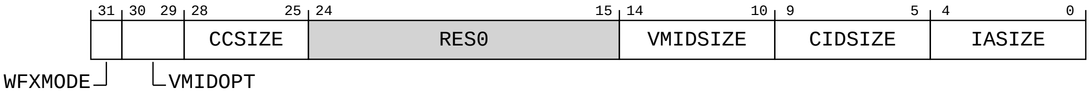


Table C-110: TRCIDR2 bit descriptions

Bits	Name	Description	Reset
[31]	WFXMODE	Indicates whether WFI and WFE instructions are classified as P0 instructions: `0` WFI and WFE instructions are not classified as P0 instructions.	
[30:29]	VMIDOPT	Indicates the options for Virtual context identifier selection. `10` Virtual context identifier selection not supported. ext-TRCCONFIGR.VMIDOPT is RES1.	
[28:25]	CCSIZE	Indicates the size of the cycle counter. `0000` The cycle counter is 12 bits in length.	
[24:15]	RES0	Reserved	`0b0`
[14:10]	VMIDSIZE	Indicates the trace unit Virtual context identifier size. `00100` 32-bit Virtual context identifier size.	

Bits	Name	Description	Reset
[9:5]	CIDSIZE	Indicates the Context identifier size. `00100` 32-bit Context identifier size.	
[4:0]	IASIZE	Virtual instruction address size. `01000` Maximum of 64-bit instruction address size.	

C.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 C-105: ext_trcidr3 bit assignments

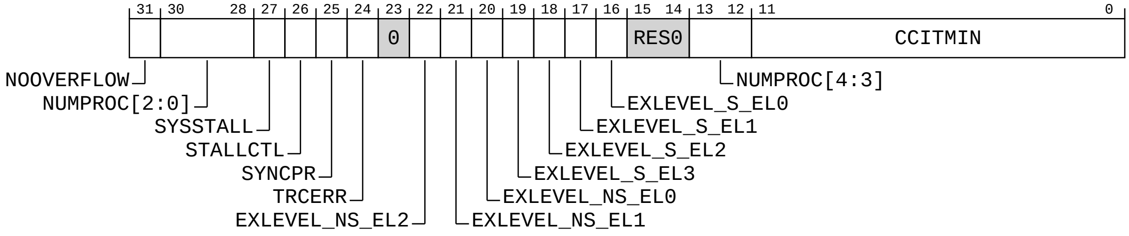


Table C-111: TRCIDR3 bit descriptions

Bits	Name	Description	Reset
[31]	NOOVERFLOW	Indicates if overflow prevention is implemented. `0` Overflow prevention is not implemented.	

Bits	Name	Description	Reset
[13:12, 30:28]	NUMPROC	Indicates the number of PEs available for tracing. **00000** The trace unit can trace one PE.	
[27]	SYSSTALL	Indicates if stalling of the PE is permitted. **1** Stalling of the PE is permitted.	
[26]	STALLCTL	Indicates if trace unit implements stalling of the PE. **1** Stalling of the PE is implemented.	
[25]	SYNCPR	Indicates if an implementation has a fixed synchronization period. **0** ext-TRCSYNCPR is read-write so software can change the synchronization period.	
[24]	TRCERR	Indicates forced tracing of System Error exceptions is implemented. **1** Forced tracing of System Error exceptions is implemented.	
[23]	RES0	Reserved	0b0
[22]	EXLEVEL_NS_EL2	Indicates if Non-secure EL2 implemented. **1** Non-secure EL2 is implemented.	
[21]	EXLEVEL_NS_EL1	Indicates if Non-secure EL1 implemented. **1** Non-secure EL1 is implemented.	
[20]	EXLEVEL_NS_EL0	Indicates if Non-secure EL0 implemented. **1** Non-secure EL0 is implemented.	
[19]	EXLEVEL_S_EL3	Indicates if Secure EL3 implemented. **1** Secure EL3 is implemented.	
[18]	EXLEVEL_S_EL2	Indicates if Secure EL2 implemented. **0** Secure EL2 is not implemented.	
[17]	EXLEVEL_S_EL1	Indicates if Secure EL1 implemented. **1** Secure EL1 is implemented.	
[16]	EXLEVEL_S_EL0	Indicates if Secure EL0 implemented. **1** Secure EL0 is implemented.	
[15:14]	RES0	Reserved	0b0

Bits	Name	Description	Reset
[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. `000000000100`	

C.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 C-106: ext_trcidr4 bit assignments

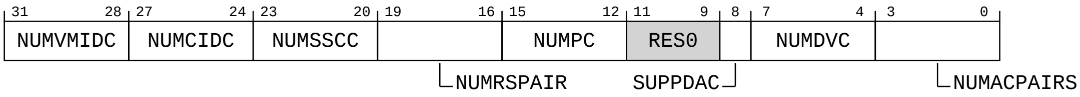


Table C-112: TRCIDR4 bit descriptions

Bits	Name	Description	Reset
[31:28]	NUMVMIDC	Indicates the number of Virtual Context Identifier Comparators that are available for tracing. `0001` The implementation has one Virtual Context Identifier Comparator.	
[27:24]	NUMCIDC	Indicates the number of Context Identifier Comparators that are available for tracing. `0001` The implementation has one Context Identifier Comparator.	

Bits	Name	Description	Reset
[23:20]	NUMSSCC	Indicates the number of Single-shot Comparator Controls that are available for tracing. **0001** The implementation has one Single-shot Comparator Control.	
[19:16]	NUMRSPAIR	Indicates the number of resource selector pairs that are available for tracing. **0111** The implementation has eight resource selector pairs.	
[15:12]	NUMPC	Indicates the number of PE Comparator Inputs that are available for tracing. **0000** No PE Comparator Inputs are available.	
[11:9]	RES0	Reserved	0b0
[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. **0** 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. **0000** No data value comparators implemented.	
[3:0]	NUMACPAIRS	Indicates the number of Address Comparator pairs that are available for tracing. **0100** The implementation has four Address Comparator pairs.	

C.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 C-107: ext_trcidr5 bit assignments

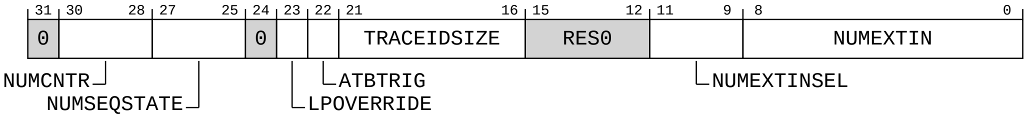


Table C-113: TRCIDR5 bit descriptions

Bits	Name	Description	Reset
[31]	RES0	Reserved	0b0
[30:28]	NUMCNTR	Indicates the number of Counters that are available for tracing. **010** Two Counters implemented.	
[27:25]	NUMSEQSTATE	Indicates if the Sequencer is implemented and the number of Sequencer states that are implemented. **100** Four Sequencer states are implemented.	
[24]	RES0	Reserved	0b0
[23]	LPOVERRIDE	Indicates support for Low-power Override Mode. **1** The trace unit supports Low-power Override Mode.	
[22]	ATBTRIG	Indicates if the implementation can support ATB triggers. **1** The implementation supports ATB triggers.	
[21:16]	TRACEIDSIZE	Indicates the trace ID width. **000111** 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. **100** 4 External Input Selector resources are available.	
[8:0]	NUMEXTIN	Indicates how many External Inputs are implemented. **111111111** Unified PMU event selection.	

C.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 C-108: ext_trcidr6 bit assignments

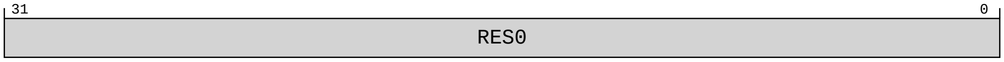


Table C-114: TRCIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.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 C-109: ext_trcidr7 bit assignments

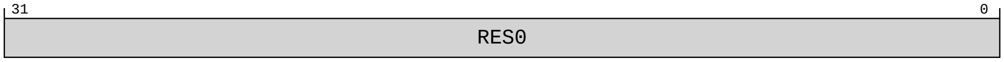


Table C-115: TRCIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.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

`0x0`

Bit descriptions

Figure C-110: ext_trcitctrl bit assignments

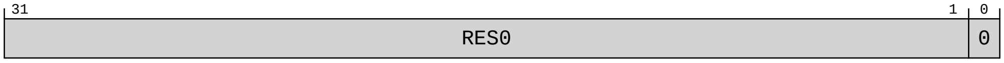


Table C-116: TRCITCTRL bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b`

C.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 C-111: ext_trcclaimset bit assignments

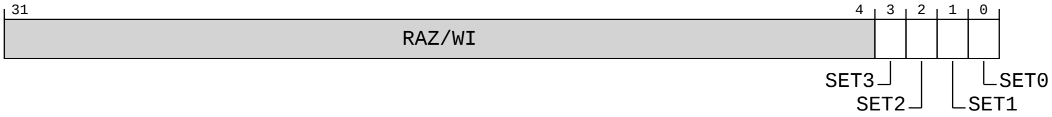


Table C-117: TRCCLAIMSET bit descriptions

Bits	Name	Description	Reset
[31:4]	RAZ/WI	Reserved	
[3]	SET3	Claim Tag Set. Indicates whether Claim Tag bit m is implemented, and is used to set Claim Tag bit m to 0b1. **0** On a read: Claim Tag bit m is not implemented. On a write: Ignored. **1** On a read: Claim Tag bit m is implemented. On a write: Set Claim Tag bit m to 0b1.	

Bits	Name	Description	Reset
[2]	SET2	Claim Tag Set. Indicates whether Claim Tag bit m is implemented, and is used to set Claim Tag bit m to 0b1. **0** On a read: Claim Tag bit m is not implemented. On a write: Ignored. **1** On a read: Claim Tag bit m is implemented. On a write: Set Claim Tag bit m to 0b1.	
[1]	SET1	Claim Tag Set. Indicates whether Claim Tag bit m is implemented, and is used to set Claim Tag bit m to 0b1. **0** On a read: Claim Tag bit m is not implemented. On a write: Ignored. **1** On a read: Claim Tag bit m is implemented. On a write: Set Claim Tag bit m to 0b1.	
[0]	SET0	Claim Tag Set. Indicates whether Claim Tag bit m is implemented, and is used to set Claim Tag bit m to 0b1. **0** On a read: Claim Tag bit m is not implemented. On a write: Ignored. **1** On a read: Claim Tag bit m is implemented. On a write: Set Claim Tag bit m to 0b1.	

C.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 C-112: ext_trcclaimclr bit assignments

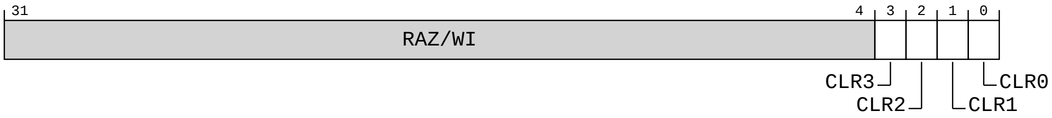


Table C-118: TRCCLAIMCLR bit descriptions

Bits	Name	Description	Reset
[31:4]	RAZ/WI	Reserved	
[3]	CLR3	Claim Tag Clear. Indicates the current status of the Claim Tag bit m, and is used to clear Claim Tag bit m to 0b0. **0** On a read: Claim Tag bit m is not set. On a write: Ignored. **1** On a read: Claim Tag bit m is set. On a write: Clear Claim tag bit m to 0b0.	
[2]	CLR2	Claim Tag Clear. Indicates the current status of the Claim Tag bit m, and is used to clear Claim Tag bit m to 0b0. **0** On a read: Claim Tag bit m is not set. On a write: Ignored. **1** On a read: Claim Tag bit m is set. On a write: Clear Claim tag bit m to 0b0.	
[1]	CLR1	Claim Tag Clear. Indicates the current status of the Claim Tag bit m, and is used to clear Claim Tag bit m to 0b0. **0** On a read: Claim Tag bit m is not set. On a write: Ignored. **1** On a read: Claim Tag bit m is set. On a write: Clear Claim tag bit m to 0b0.	

Bits	Name	Description	Reset
[0]	CLR0	Claim Tag Clear. Indicates the current status of the Claim Tag bit m, and is used to clear Claim Tag bit m to 0b0. **0** On a read: Claim Tag bit m is not set. On a write: Ignored. **1** On a read: Claim Tag bit m is set. On a write: Clear Claim tag bit m to 0b0.	

C.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 C-113: ext_trcdevarch bit assignments

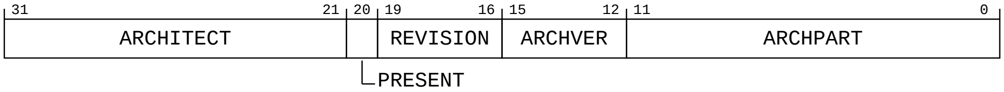


Table C-119: TRCDEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	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. **`01000111011`** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited. Other values are defined by the JEDEC JEP106 standard. This field reads as 0x23B.	
[20]	PRESENT	DEVARCH Present. Defines that the DEVARCH register is present. **`1`** Device Architecture information present.	
[19:16]	REVISION	Revision. Defines the architecture revision of the component. **`0000`** ETE Version 1.0. All other values are reserved.	
[15:12]	ARCHVER	Architecture Version. Defines the architecture version of the component. **`0101`** ETE Version 1. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHVER is ARCHID[15:12]. This field reads as 0x5.	
[11:0]	ARCHPART	Architecture Part. Defines the architecture of the component. **`101000010011`** Arm PE trace architecture. ARCHVER and ARCHPART are also defined as a single field, ARCHID, so that ARCHPART is ARCHID[11:0]. This field reads as 0xA13.	

C.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 C-114: ext_trcdevid2 bit assignments

Table C-120: TRCDEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

C.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 C-115: ext_trcdevid1 bit assignments

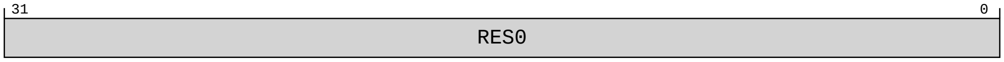


Table C-121: TRCDEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.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 C-116: ext_trcdevid bit assignments

Table C-122: TRCDEVID bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.6.24 TRCDEVTYPE, Device Type Register

Provides discovery information for the component. If the part number field is not recognized, a debugger can report the information that is provided by TRCDEVTYPE about the component instead.

For additional information see the CoreSight Architecture Specification.

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 C-117: ext_trcdevtype bit assignments

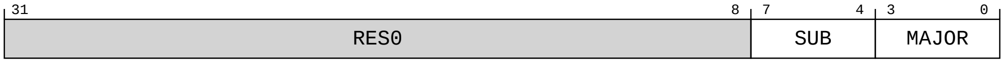


Table C-123: TRCDEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:4]	SUB	Component sub-type. **`0001`** When MAJOR == 0x3 (Trace source): Associated with a PE. This field reads as 0x1.	
[3:0]	MAJOR	Component major type. **`0011`** Trace source. Other values are defined by the CoreSight Architecture. This field reads as 0x3.	

C.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 C-118: ext_trcpidr4 bit assignments

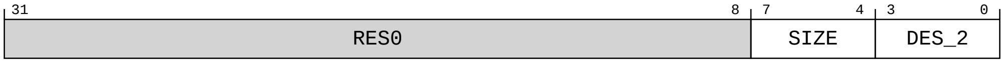


Table C-124: TRCPIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	SIZE	Size of the component. The distance from the start of the address space used by this component to the end of the component identification registers. A value of 0b0000 means one of the following is true: • The component uses a single 4KB block. • The component uses an **IMPLEMENTATION DEFINED** number of 4KB blocks. Any other value means the component occupies $2^{\text{TRCPIDR4.SIZE}}$ 4KB blocks. Using this field to indicate the size of the component is deprecated. This field might not correctly indicate the size of the component. Arm recommends that software determine the size of the component from the Unique Component Identifier fields, and other **IMPLEMENTATION DEFINED** registers in the component. This field reads as 0b0000.	

Bits	Name	Description	Reset
[3:0]	DES_2	Designer, JEP106 continuation code. This is the JEDEC-assigned JEP106 bank identifier for the designer of the component, minus 1. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. This field reads as an **IMPLEMENTATION DEFINED** value. **Note:** For a component designed by Arm Limited, the JEP106 bank is 5, meaning this field has the value 0x4.	

C.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 C-119: ext_trcpidr5 bit assignments

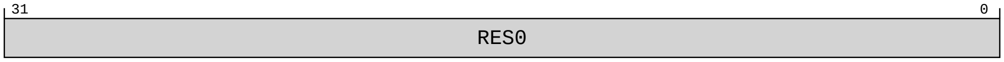


Table C-125: TRCPIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

C.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 C-120: ext_trcpidr6 bit assignments

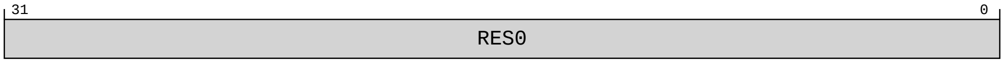


Table C-126: TRCPIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.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 C-121: ext_trcpidr7 bit assignments

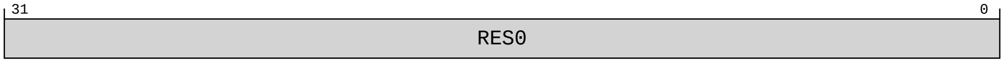


Table C-127: TRCPIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	0b0

C.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 C-122: ext_trcpidr0 bit assignments

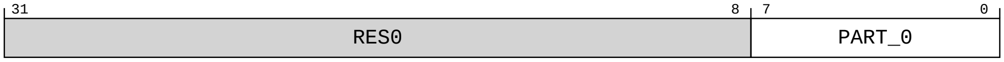


Table C-128: TRCPIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PART_0	Part number, bits [7:0]. The part number is selected by the designer of the component, and is stored in ext-TRCPIDR1.PART_1 and TRCPIDR0.PART_0. This field reads as an **IMPLEMENTATION DEFINED** value.	

C.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 C-123: ext_trcpidr1 bit assignments

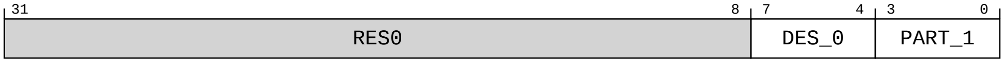


Table C-129: TRCPIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:4]	DES_0	Designer, JEP106 identification code, bits [3:0]. TRCPIDR1.DES_0 and ext-TRCPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. This field reads as an **IMPLEMENTATION DEFINED** value. **Note:** For a component designed by Arm Limited, the JEP106 identification code is 0x3B.	
[3:0]	PART_1	Part number, bits [11:8]. The part number is selected by the designer of the component, and is stored in TRCPIDR1.PART_1 and ext-TRCPIDR0.PART_0. This field reads as an **IMPLEMENTATION DEFINED** value.	

C.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 C-124: ext_trcpidr2 bit assignments

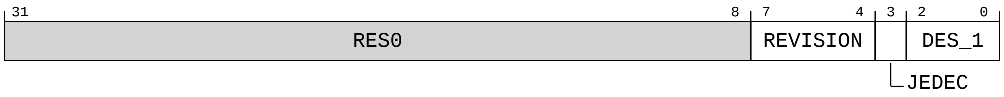


Table C-130: TRCPIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	REVISION	Component major revision. TRCPIDR2.REVISION and ext-TRCPIDR3.REVAND together form the revision number of the component, with TRCPIDR2.REVISION being the most significant part and ext-TRCPIDR3.REVAND the least significant part. When a component is changed, TRCPIDR2.REVISION or ext-TRCPIDR3.REVAND must be increased to ensure that software can differentiate the different revisions of the component. If TRCPIDR2.REVISION is increased then ext-TRCPIDR3.REVAND should be set to 0b0000. This field reads as an **IMPLEMENTATION DEFINED** value.	
[3]	JEDEC	JEDEC-assigned JEP106 implementer code is used. This bit reads as one.	
[2:0]	DES_1	Designer, JEP106 identification code, bits [6:4]. ext-TRCPIDR1.DES_0 and TRCPIDR2.DES_1 together form the JEDEC-assigned JEP106 identification code for the designer of the component. The parity bit in the JEP106 identification code is not included. The code identifies the designer of the component, which might not be not the same as the implementer of the device containing the component. To obtain a number, or to see the assignment of these codes, contact JEDEC http://www.jedec.org. This field reads as an **IMPLEMENTATION DEFINED** value. **Note:** For a component designed by Arm Limited, the JEP106 identification code is 0x3B.	

C.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 C-125: ext_trcpidr3 bit assignments

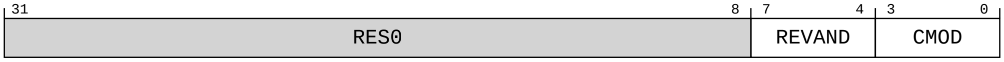


Table C-131: TRCPIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:4]	REVAND	Component minor revision. ext-TRCPIDR2.REVISION and TRCPIDR3.REVAND together form the revision number of the component, with ext-TRCPIDR2.REVISION being the most significant part and TRCPIDR3.REVAND the least significant part. When a component is changed, ext-TRCPIDR2.REVISION or TRCPIDR3.REVAND must be increased to ensure that software can differentiate the different revisions of the component. If ext-TRCPIDR2.REVISION is increased then TRCPIDR3.REVAND should be set to 0b0000. This field reads as an **IMPLEMENTATION DEFINED** value.	
[3:0]	CMOD	Customer Modified. Indicates the component has been modified. A value of 0b0000 means the component is not modified from the original design. Any other value means the component has been modified in an **IMPLEMENTATION DEFINED** way. For any two components with the same Unique Component Identifier: • If the value of the CMOD fields of both components equals zero, the components are identical. • If the CMOD fields of both components have the same non-zero value, it does not necessarily mean that they have the same modifications. • If the value of the CMOD field of either of the two components is non-zero, they might not be identical, even though they have the same Unique Component Identifier. This field reads as an **IMPLEMENTATION DEFINED** value.	

C.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 C-126: ext_trccidr0 bit assignments

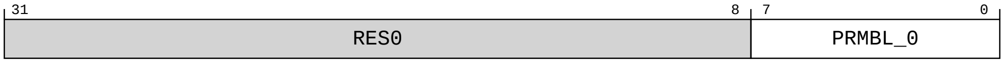


Table C-132: TRCCIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_0	Component identification preamble, segment 0. This field reads as 0x0D.	

C.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 C-127: ext_trccidr1 bit assignments

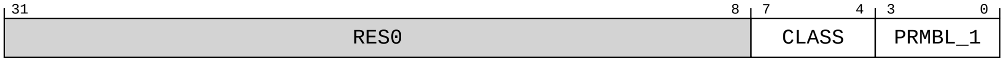


Table C-133: TRCCIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:4]	CLASS	Component class. **`1001`** CoreSight peripheral. Other values are defined by the CoreSight Architecture. This field reads as 0x9.	
[3:0]	PRMBL_1	Component identification preamble, segment 1. This field reads as 0x0.	

C.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 C-128: ext_trccidr2 bit assignments

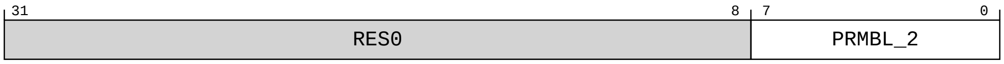


Table C-134: TRCCIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:0]	PRMBL_2	Component identification preamble, segment 2. This field reads as 0x05.	

C.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 C-129: ext_trccidr3 bit assignments

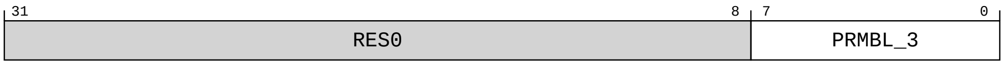


Table C-135: TRCCIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`

Bits	Name	Description	Reset
[7:0]	PRMBL_3	Component identification preamble, segment 3. This field reads as 0xB1.	

C.7 ROM table register summary

The summary table provides an overview of memory-mapped ROM table registers in the Cortex®-A510 core. Individual register descriptions provide detailed information.

Table C-136: ROM table register summary

Offset	Name	Reset	Width	Description
`0x0`	ROMENTRY0	See individual bit resets	32-bit	Class `0x9` ROM table entries
`0x4`	ROMENTRY1	See individual bit resets	32-bit	Class `0x9` ROM table entries
`0x8`	ROMENTRY2	See individual bit resets	32-bit	Class `0x9` ROM table entries
`0xC`	ROMENTRY3	See individual bit resets	32-bit	Class `0x9` ROM table entries
`0x10`	ROMENTRY4	See individual bit resets	32-bit	Class `0x9` ROM table entries
`0x14`	ROMENTRY5	See individual bit resets	32-bit	Class `0x9` ROM table entries
`0x18`	ROMENTRY6	See individual bit resets	32-bit	Class `0x9` ROM table entries
`0x1C`	ROMENTRY7	See individual bit resets	32-bit	Class `0x9` ROM table entries
`0xFBC`	DEVARCH	See individual bit resets	32-bit	Device Architecture Register
`0xFC0`	DEVID2	`0x0`	32-bit	Device Configuration Register 2
`0xFC4`	DEVID1	`0x0`	32-bit	Device Configuration Register 1
`0xFC8`	DEVID	See individual bit resets	32-bit	Device Configuration Register
`0xFCC`	DEVTYPE	See individual bit resets	32-bit	Device Type Register
`0xFD0`	PIDR4	See individual bit resets	32-bit	Peripheral Identification Register 4
`0xFD4`	PIDR5	`0x0`	32-bit	Peripheral Identification Register 5
`0xFD8`	PIDR6	`0x0`	32-bit	Peripheral Identification Register 6
`0xFDC`	PIDR7	`0x0`	32-bit	Peripheral Identification Register 7
`0xFE0`	PIDR0	See individual bit resets	32-bit	Peripheral Identification Register 0
`0xFE4`	PIDR1	See individual bit resets	32-bit	Peripheral Identification Register 1
`0xFE8`	PIDR2	See individual bit resets	32-bit	Peripheral Identification Register 2
`0xFEC`	PIDR3	See individual bit resets	32-bit	Peripheral Identification Register 3
`0xFF0`	CIDR0	See individual bit resets	32-bit	Component Identification Register 0
`0xFF4`	CIDR1	See individual bit resets	32-bit	Component Identification Register 1
`0xFF8`	CIDR2	See individual bit resets	32-bit	Component Identification Register 2
`0xFFC`	CIDR3	See individual bit resets	32-bit	Component Identification Register 3

Related information

16.5 ROM table on page 96

C.7.1 ROMENTRY0, Class 0x9 ROM Table Entries

A Class `0x9` ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component _n_, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class `0x9` ROM Table.

The first entry, entry 0, has offset `0x000`, then ext-ROMENTRY<n> has the offset `0x000` + _n_×4, where 0 ≤ _n_ ≤ 511.

If the number of components, _N_, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:

- ROM Table entries representing components have offsets from `0x000` to (_N_-1)×4.
- The ext-ROMENTRY<n> at offset _N_×4, which has a PRESENT field with the value `0b00`, indicates the end of the ROM Table.

If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from `0x000` to `0x7FC`. If a ROM Table entry is present at offset `0x7FC`, its PRESENT field must have a value of either `0b00` or `0b11`, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value `0b11`.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0x0

Reset value

See individual bit resets.

Bit descriptions

Figure C-130: ext_romentry0 bit assignments

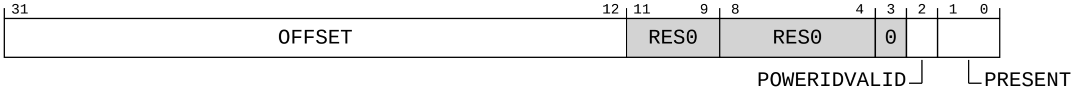


Table C-137: ROMENTRY0 bit descriptions

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). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **00000000000000010000** Core 0 Debug	
[11:3]	RES0	Reserved	0b0
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0** A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **11** The ROM Entry is present.	

C.7.2 ROMENTRY1, Class 0x9 ROM Table Entries

A Class 0x9 ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component _n_, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class 0x9 ROM Table.

The first entry, entry 0, has offset 0x000, then ext-ROMENTRY<n> has the offset 0x000 + _n_×4, where 0 ≤ _n_ ≤ 511.

If the number of components, _N_, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:

- ROM Table entries representing components have offsets from 0x000 to (_N_-1)×4.
- The ext-ROMENTRY<n> at offset _N_×4, which has a PRESENT field with the value 0b00, indicates the end of the ROM Table.

If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from `0x000` to `0x7FC`. If a ROM Table entry is present at offset `0x7FC`, its PRESENT field must have a value of either `0b00` or `0b11`, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value `0b11`.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0x4

Reset value

See individual bit resets.

Bit descriptions

Figure C-131: ext_romentry1 bit assignments

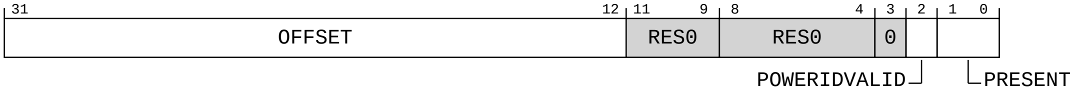


Table C-138: ROMENTRY1 bit descriptions

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). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **00000000000000100000** Core 0 PMU	
[11:3]	RES0	Reserved	`0b0`

Bits	Name	Description	Reset
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. `0` A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. `11` The ROM Entry is present.	

C.7.3 ROMENTRY2, Class 0x9 ROM Table Entries

A Class `0x9` ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component _n_, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class `0x9` ROM Table.

The first entry, entry 0, has offset `0x000`, then ext-ROMENTRY<n> has the offset `0x000` + _n_×4, where 0 ≤ _n_ ≤ 511.

If the number of components, _N_, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:

- ROM Table entries representing components have offsets from `0x000` to (_N_-1)×4.
- The ext-ROMENTRY<n> at offset _N_×4, which has a PRESENT field with the value `0b00`, indicates the end of the ROM Table.

If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from `0x000` to `0x7FC`. If a ROM Table entry is present at offset `0x7FC`, its PRESENT field must have a value of either `0b00` or `0b11`, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value `0b11`.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0x8

Reset value

See individual bit resets.

Bit descriptions

Figure C-132: ext_romentry2 bit assignments

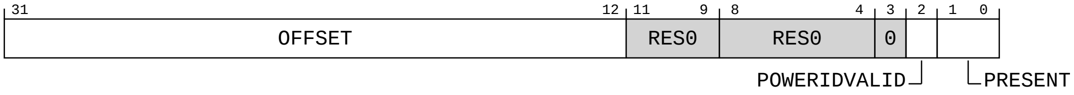


Table C-139: ROMENTRY2 bit descriptions

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). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **00000000000000110000** Core 0 ETM	
[11:3]	RES0	Reserved	`0b0`
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0** A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **11** The ROM Entry is present.	

C.7.4 ROMENTRY3, Class 0x9 ROM Table Entries

A Class `0x9` ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component _n_, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class `0x9` ROM Table.

The first entry, entry 0, has offset `0x000`, then ext-ROMENTRY<n> has the offset `0x000` + _n_×4, where 0 ≤ _n_ ≤ 511.
If the number of components, _N_, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:

- ROM Table entries representing components have offsets from `0x000` to (_N_-1)×4.

- The ext-ROMENTRY<n> at offset _N_×4, which has a PRESENT field with the value `0b00`, indicates the end of the ROM Table.

If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from `0x000` to `0x7FC`. If a ROM Table entry is present at offset `0x7FC`, its PRESENT field must have a value of either `0b00` or `0b11`, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value `0b11`.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xC

Reset value

See individual bit resets.

Bit descriptions

Figure C-133: ext_romentry3 bit assignments

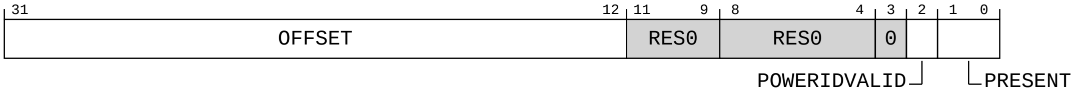


Table C-140: ROMENTRY3 bit descriptions

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). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **00000000000001000000** ELA	

Bits	Name	Description	Reset
[11:3]	RES0	Reserved	`0b0`
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **`0`** A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **`10`** The ROM entry is not present, and this ext-ROMENTRY<n> is not the final entry in a ROM Table with fewer than the maximum number of entries. If PRESENT has this value, all other fields in this entry are UNKNOWN. **`11`** The ROM Entry is present.	

C.7.5 ROMENTRY4, Class 0x9 ROM Table Entries

A Class `0x9` ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component _n_, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class `0x9` ROM Table.

The first entry, entry 0, has offset `0x000`, then ext-ROMENTRY<n> has the offset `0x000` + _n_×4, where 0 ≤ _n_ ≤ 511.

If the number of components, _N_, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:

- ROM Table entries representing components have offsets from `0x000` to (_N_-1)×4.
- The ext-ROMENTRY<n> at offset _N_×4, which has a PRESENT field with the value `0b00`, indicates the end of the ROM Table.

If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from `0x000` to `0x7FC`. If a ROM Table entry is present at offset `0x7FC`, its PRESENT field must have a value of either `0b00` or `0b11`, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value `0b11`.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0x10

Reset value

See individual bit resets.

Bit descriptions

Figure C-134: ext_romentry4 bit assignments

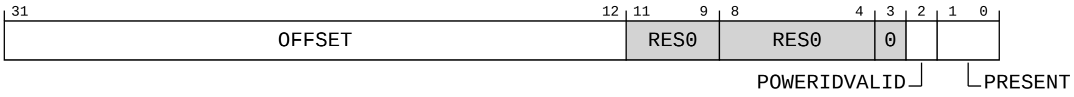


Table C-141: ROMENTRY4 bit descriptions

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). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **00000000000100010000** Core 1 Debug	
[11:3]	RES0	Reserved	0b0
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0** A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **00** The ROM entry is not present, and this ext-ROMENTRY<n> is the final entry in the ROM Table. If PRESENT has this value, all other fields in this ext-ROMENTRY<n> must be zero. **11** The ROM Entry is present.	

C.7.6 ROMENTRY5, Class 0x9 ROM Table Entries

A Class `0x9` ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component _n_, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class `0x9` ROM Table.

The first entry, entry 0, has offset `0x000`, then ext-ROMENTRY<n> has the offset `0x000` + _n_×4, where 0 ≤ _n_ ≤ 511.

If the number of components, _N_, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:

- ROM Table entries representing components have offsets from `0x000` to (_N_-1)×4.
- The ext-ROMENTRY<n> at offset _N_×4, which has a PRESENT field with the value `0b00`, indicates the end of the ROM Table.

If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from `0x000` to `0x7FC`. If a ROM Table entry is present at offset `0x7FC`, its PRESENT field must have a value of either `0b00` or `0b11`, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value `0b11`.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0x14

Reset value

See individual bit resets.

Bit descriptions

Figure C-135: ext_romentry5 bit assignments

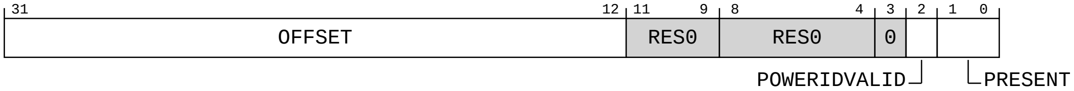


Table C-142: ROMENTRY5 bit descriptions

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). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **00000000000100100000** Core 1 PMU	
[11:3]	RES0	Reserved	0b0
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0** A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **00** The ROM entry is not present, and this ext-ROMENTRY<n> is the final entry in the ROM Table. If PRESENT has this value, all other fields in this ext-ROMENTRY<n> must be zero. **11** The ROM Entry is present.	

C.7.7 ROMENTRY6, Class 0x9 ROM Table Entries

A Class `0x9` ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component _n_, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class `0x9` ROM Table.

The first entry, entry 0, has offset `0x000`, then ext-ROMENTRY<n> has the offset `0x000` + _n_×4, where 0 ≤ _n_ ≤ 511.

If the number of components, _N_, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:

- ROM Table entries representing components have offsets from `0x000` to (_N_-1)×4.

- The ext-ROMENTRY<n> at offset _N_×4, which has a PRESENT field with the value `0b00`, indicates the end of the ROM Table.

If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from `0x000` to `0x7FC`. If a ROM Table entry is present at offset `0x7FC`, its PRESENT field must have a value of either `0b00` or `0b11`, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value `0b11`.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0x18

Reset value

See individual bit resets.

Bit descriptions

Figure C-136: ext_romentry6 bit assignments

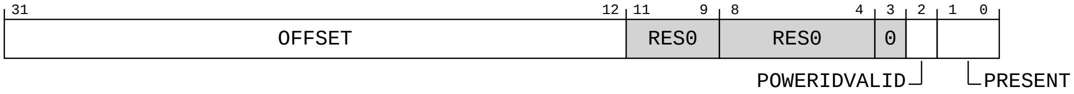


Table C-143: ROMENTRY6 bit descriptions

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). If a component occupies more than a single 4KB block, OFFSET points to the 4KB block which contains the Peripheral ID and Component ID registers for the component. Negative values of OFFSET are permitted, using two's complement. **00000000000100110000** Core 1 ETM	

Bits	Name	Description	Reset
[11:3]	RES0	Reserved	0b0
[2]	POWERIDVALID	Indicates if the Power domain ID field contains a Power domain ID. **0** A power domain ID is not provided.	
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. **00** The ROM entry is not present, and this ext-ROMENTRY<n> is the final entry in the ROM Table. If PRESENT has this value, all other fields in this ext-ROMENTRY<n> must be zero. **11** The ROM Entry is present.	

C.7.8 ROMENTRY7, Class 0x9 ROM Table Entries

A Class 0x9 ROM Table contains up to 512 ROM Table entries. Each entry that is present, ext-ROMENTRY<n>, provides the address offset of the address space of one CoreSight component, component _n_, along with information about its power domain.

The series of ROM Table entries starts at the base address of the Class 0x9 ROM Table.

The first entry, entry 0, has offset 0x000, then ext-ROMENTRY<n> has the offset 0x000 + _n_×4, where 0 ≤ _n_ ≤ 511.

If the number of components, _N_, is lower than the maximum supported number, 512, the offsets of the ROM Table entries are in the following range:

- ROM Table entries representing components have offsets from 0x000 to (_N_-1)×4.
- The ext-ROMENTRY<n> at offset _N_×4, which has a PRESENT field with the value 0b00, indicates the end of the ROM Table.

If the number of components is equal to the maximum supported number, the ROM Table entries have offsets from 0x000 to 0x7FC. If a ROM Table entry is present at offset 0x7FC, its PRESENT field must have a value of either 0b00 or 0b11, and it must be interpreted as the final entry of the ROM Table, even if its PRESENT field has the value 0b11.

A component that requires differentiation between external and internal accesses may provide two views, one for internal and one for external accesses. For these components, Arm strongly recommends that ROM Tables provide a pointer only to the external view.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0x1C

Reset value

See individual bit resets.

Bit descriptions

Figure C-137: ext_romentry7 bit assignments

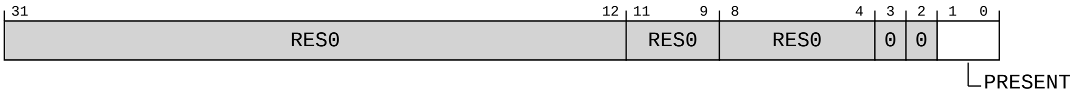


Table C-144: ROMENTRY7 bit descriptions

Bits	Name	Description	Reset
[31:2]	RES0	Reserved	`0b0`
[1:0]	PRESENT	Indicates whether an entry is present at this location in the ROM Table. `00` The ROM entry is not present, and this ext-ROMENTRY<n> is the final entry in the ROM Table. If PRESENT has this value, all other fields in this ext-ROMENTRY<n> must be zero.	

C.7.9 DEVARCH, 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

ROM table

Register offset

0xFBC

Reset value

See individual bit resets.

Bit descriptions

Figure C-138: ext_devarch bit assignments

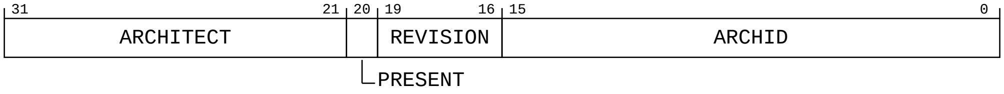


Table C-145: DEVARCH bit descriptions

Bits	Name	Description	Reset
[31:21]	ARCHITECT	Architect. **`01000111011`** JEP106 continuation code 0x4, ID code 0x3B. Arm Limited.	
[20]	PRESENT	Present. **`1`** DEVARCH information present.	
[19:16]	REVISION	Revision. **`0000`** Revision 0.	
[15:0]	ARCHID	Architecture ID. **`0000101011110111`** ROM Table v0. The debug tool must inspect ext-DEVTYPE and ext-DEVID to determine further information about the ROM Table.	

C.7.10 DEVID2, Device Configuration Register 2

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFC0

Reset value

`0x0`

Bit descriptions

Figure C-139: ext_devid2 bit assignments

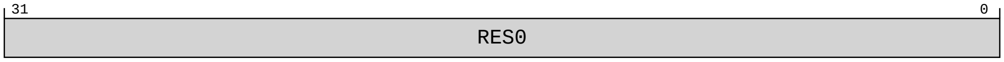


Table C-146: DEVID2 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.7.11 DEVID1, Device Configuration Register 1

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFC4

Reset value

`0x0`

Bit descriptions

Figure C-140: ext_devid1 bit assignments

Table C-147: DEVID1 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.7.12 DEVID, Device Configuration Register

Indicates the capabilities of the component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFC8

Reset value

See individual bit resets.

Bit descriptions

Figure C-141: ext_devid bit assignments

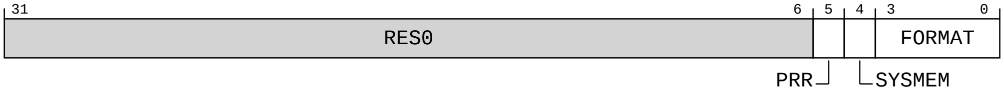


Table C-148: DEVID bit descriptions

Bits	Name	Description	Reset
[31:6]	RES0	Reserved	0b0
[5]	PRR	Power Request functionality included. **0** Power Request functionality not included. If any ROM Table entries contain power domain IDs, a GPR must be present, and pointed to by the ROM Table. The GPR provides functionality to control the power domains. ext-PRIDR0 is not implemented.	
[4]	SYSMEM	System memory present. Indicates whether system memory is present on the bus that connects to the ROM Table. **0** System memory is not present on the bus. This value indicates that the bus is a dedicated debug bus. The ROM Table indicates all the valid addresses in the memory system that the ADI is connected to, and the result of accessing any other address is UNPREDICTABLE.	

Bits	Name	Description	Reset
[3:0]	FORMAT	ROM format. `0000` 32-bit format 0.	

C.7.13 DEVTYPE, 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

ROM table

Register offset

0xFCC

Reset value

See individual bit resets.

Bit descriptions

Figure C-142: ext_devtype bit assignments

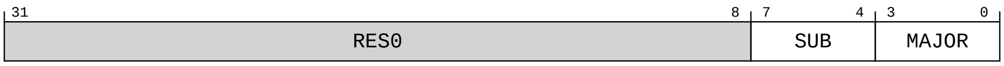


Table C-149: DEVTYPE bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:4]	SUB	Sub number `0000` Other, undefined.	
[3:0]	MAJOR	Major number `0000` Miscellaneous.	

C.7.14 PIDR4, Peripheral Identification Register 4

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFD0

Reset value

See individual bit resets.

Bit descriptions

Figure C-143: ext_pidr4 bit assignments

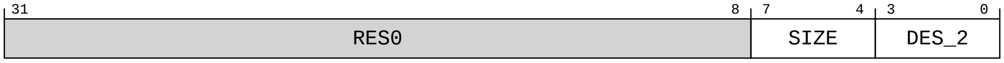


Table C-150: PIDR4 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:4]	SIZE	Size of the component. RAZ. Log_2 of the number of 4KB pages from the start of the component to the end of the component ID registers. **`0000`** A ROM Table occupies a single 4KB block of memory.	
[3:0]	DES_2	Designer, JEP106 continuation code, least significant nibble. For Arm Limited, this field is 0b0100. **`0100`** Arm Limited	

C.7.15 PIDR5, Peripheral Identification Register 5

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFD4

Reset value

`0x0`

Bit descriptions

Figure C-144: ext_pidr5 bit assignments

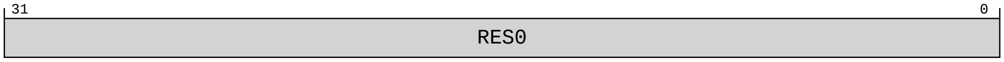


Table C-151: PIDR5 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.7.16 PIDR6, Peripheral Identification Register 6

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFD8

Reset value

`0x0`

Bit descriptions

Figure C-145: ext_pidr6 bit assignments

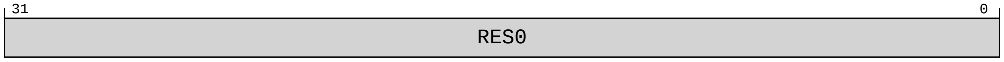


Table C-152: PIDR6 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.7.17 PIDR7, Peripheral Identification Register 7

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFDC

Reset value

`0x0`

Bit descriptions

Figure C-146: ext_pidr7 bit assignments

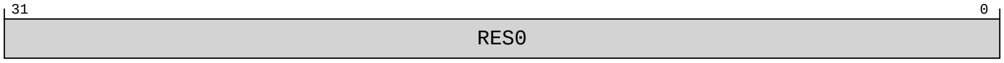


Table C-153: PIDR7 bit descriptions

Bits	Name	Description	Reset
[31:0]	RES0	Reserved	`0b0`

C.7.18 PIDR0, Peripheral Identification Register 0

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFE0

Reset value

See individual bit resets.

Bit descriptions

Figure C-147: ext_pidr0 bit assignments

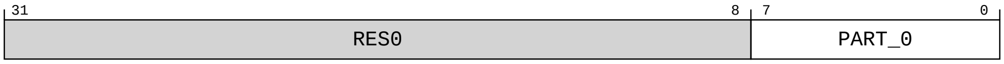


Table C-154: PIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:0]	PART_0	Part number, least significant byte. **`01000110`** Cortex®-A510	

C.7.19 PIDR1, Peripheral Identification Register 1

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFE4

Reset value

See individual bit resets.

Bit descriptions

Figure C-148: ext_pidr1 bit assignments

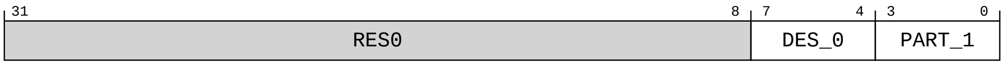


Table C-155: PIDR1 bit descriptions

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. **`1011`** Arm Limited	
[3:0]	PART_1	Part number, most significant nibble. **`1101`** Cortex®-A510	

C.7.20 PIDR2, Peripheral Identification Register 2

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFE8

Reset value

See individual bit resets.

Bit descriptions

Figure C-149: ext_pidr2 bit assignments

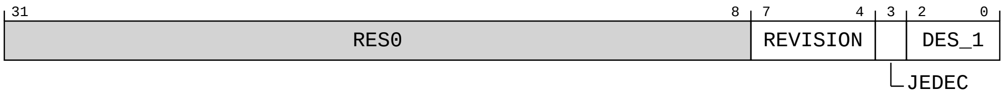


Table C-156: PIDR2 bit descriptions

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. **0000** r0p3	
[3]	JEDEC	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. **011** Arm Limited	

C.7.21 PIDR3, Peripheral Identification Register 3

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFEC

Reset value

See individual bit resets.

Bit descriptions

Figure C-150: ext_pidr3 bit assignments

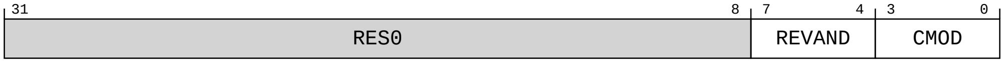


Table C-157: PIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:4]	REVAND	Part minor revision. Parts using ext-PIDR2.REVISION as an extension to the Part number must use this field as a major revision number. `0011` r0p3	
[3:0]	CMOD	Customer modified. Indicates someone other than the Designer has modified the component. `0000`	

C.7.22 CIDR0, Component Identification Register 0

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFF0

Reset value

See individual bit resets.

Bit descriptions

Figure C-151: ext_cidr0 bit assignments

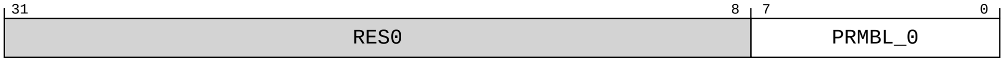


Table C-158: CIDR0 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_0	CoreSight component identification preamble. `00001101` CoreSight component identification preamble.	

C.7.23 CIDR1, Component Identification Register 1

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFF4

Reset value

See individual bit resets.

Bit descriptions

Figure C-152: ext_cidr1 bit assignments

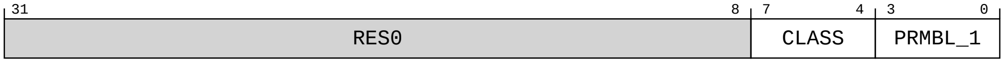


Table C-159: CIDR1 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:4]	CLASS	CoreSight component class. **`1001`** CoreSight component.	
[3:0]	PRMBL_1	CoreSight component identification preamble. **`0000`** CoreSight component identification preamble.	

C.7.24 CIDR2, Component Identification Register 2

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFF8

Reset value

See individual bit resets.

Bit descriptions

Figure C-153: ext_cidr2 bit assignments

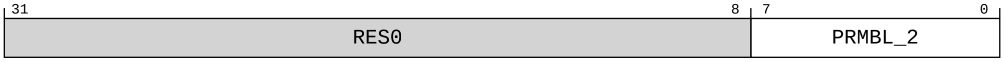


Table C-160: CIDR2 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	0b0
[7:0]	PRMBL_2	CoreSight component identification preamble. **00000101** CoreSight component identification preamble.	

C.7.25 CIDR3, Component Identification Register 3

Provide information to identify a CoreSight component.

Configurations

This register is available in all configurations.

Attributes

Width

32

Functional group

ROM table

Register offset

0xFFC

Reset value

See individual bit resets.

Bit descriptions

Figure C-154: ext_cidr3 bit assignments

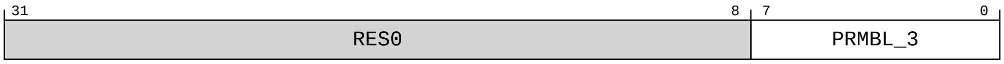


Table C-161: CIDR3 bit descriptions

Bits	Name	Description	Reset
[31:8]	RES0	Reserved	`0b0`
[7:0]	PRMBL_3	CoreSight component identification preamble. **`10110001`** CoreSight component identification preamble.	

Appendix D Document revisions

This appendix records the changes between released issues of this document.

D.1 Revisions

Changes between released issues of this book are summarized in tables.

Table D-1: Issue 0000-01

Change	Location
First Confidential alpha release for r0p0	-

Table D-2: Differences between issue 0000-01 and issue 0000-02

Change	Location
First Confidential beta release for r0p0	-
Various additions and clarifications	2 The Cortex-A510 core on page 22
Various additions and clarifications	3 Technical overview on page 32
Minor edits	4 Clocks and resets on page 38
Edits to diagrams and other clarifications	5.1 Voltage and power domains on page 39
Various additions and clarifications	5.2 Architectural clock gating modes on page 43
Added Warm reset to power modes table	5.4 Core power modes on page 46
Clarified description	5.4.2 Off mode on page 48
Clarified description	5.4.4 Functional retention mode on page 48
Clarified description	5.4.5 Full retention mode on page 49
Various additions and clarifications	5.5 Complex power modes on page 50
Various additions and clarifications to chapter, including new information about *Common not Private* (CnP) and new section TLB match process	6 Memory management on page 54
Clarified description of L1 instruction memory system features	7 L1 instruction memory system on page 62
Various additions and clarifications to chapter, including data cache behavior, transient memory, and non-temporal loads	8 L1 data memory system on page 65
Added a note about Direct connect feature, and clarified description of the optional integrated L2 cache	9 L2 memory system on page 72
Added new chapter	10 Direct access to internal memory on page 76
Clarified description of uncorrected faults	11.3 Fault detection and reporting on page 81
Added new registers	15 System control on page 89
Added APB memory-mapped interface to description	18.5 Embedded Trace Macrocell register interfaces on page 121

Table D-3: Differences between issue 0000-02 and issue 0000-08

Change	Location
First Confidential limited access release for r0p0	-
Minor modifications to introduction text and figures	2 The Cortex-A510 core on page 22
Minor modifications to features lists	2.1 Cortex-A510 core features on page 23
Added L2 cache, L2 slices, L2 cache data RAM partitions, and Evict/Allocate feature parameters	2.2 Cortex-A510 core implementation options on page 24
Clarified status description of Arm®v9.0-A SVE2 support	2.4 Supported standards and specifications on page 26
Clarified sentence about using EDA tool	2.5 Test features on page 29
Added L1 instruction memory system, *TRace Buffer Extension* (TRBE), L2 cache, and *Embedded Logic Analyzer* (ELA) to components list	3 Technical overview on page 32
Modified introduction text to refer to a complex rather than a core.	3.1 Core Components on page 32
Modified the components block diagram	
Clarified descriptions of various components throughout section	
Clarified descriptions of clock gating and Warm reset	4 Clocks and resets on page 38
Modified introduction text to refer to a complex rather than a core	5 Power management on page 39
Various clarifications, including addition of separate voltage and power domain figures	5.1 Voltage and power domains on page 39
Minor clarifications	5.2.1 Wait for Interrupt and Wait for Event on page 43
Minor clarifications	5.2.2 Low-power state behavior considerations on page 44
Clarified the applicability to cores and complexes. Other minor clarifications.	5.3 Power control on page 44
Clarified the applicability to the shared logic and other minor clarifications	5.4 Core power modes on page 46
Clarified descriptions of Off mode, Emulated off mode, Full retention mode, and Debug recovery mode	
Minor clarifications to introduction text	5.5 Complex power modes on page 50
Renamed section and added minor clarifications	5.6 Cortex-A510 core powerup and powerdown sequence on page 52
Minor clarifications	6.1 Memory Management Unit components on page 54
Modifications to list of translation regimes and list of conditions	6.3 Translation Lookaside Buffer match process on page 56
Minor clarifications	6.4 Translation table walks on page 57
Clarified description of External aborts and Conflict aborts	6.6 Responses on page 59
Minor clarifications	6.7 Memory behavior and supported memory types on page 60
Minor modifications to introduction text and table	7 L1 instruction memory system on page 62
Minor clarifications	7.3 Program flow prediction on page 63
Various clarifications	8.1 L1 data cache behavior on page 65
Renamed atomic instructions subsection and modified description. Clarified Non-temporal loads description.	8.3 Memory system implementation on page 67
Minor clarifications	8.5 Data prefetching on page 70
Minor modifications to introduction text and table	9 L2 memory system on page 72

Change	Location
Added a note about allocations to L2 cache and other minor clarifications.	9.1 Optional integrated L2 cache on page 73
Added new section	9.3 Transaction capabilities on page 74
Major revisions	10 Direct access to internal memory on page 76
Added cache protection information to introduction text	11 RAS extension support on page 79
Minor clarifications to introduction text and modified subsection titles	11.3 Fault detection and reporting on page 81
Modified list of error detection and reporting registers	11.4 Error detection and reporting on page 81
Modified description of error reporting and performance monitoring	
Added information about different error types	11.5 Error injection on page 82
Added register summary table	11.7 RAS registers 2 on page 83
Added register summary table	12.2 GIC register summary on page 86
Minor modifications to introduction text	14 Scalable Vector Extensions support on page 88
Modified debug register core interface introduction text and added information about ELA registers	16 Debug on page 91
Added new section	16.5 ROM table on page 96
Added new section	16.7 ROM table register summary on page 97
Added performance events tables and register summary	17 Performance Monitors Extension support on page 99
Added new section	18.7 Embedded Trace Extension events on page 122
Added new section	18.8 ETE register summary on page 122
Added register summary table	19.3 TRBE register summary on page 124
Minor correction	20.2 Activity monitors counters on page 126
Added new section	20.4 Activity monitors register summary on page 127
Added new section	20.5 Memory-mapped AMU register summary on page 127
Added new appendix	B AArch64 registers on page 130
Added new appendix	C External registers on page 379

Table D-4: Differences between issue 0000-08 and issue 0001-10

Change	Location
First Confidential early access release for r0p1	-
Added a note about cache indices	2.2 Cortex-A510 core implementation options on page 24
Changed voltage domain names to VCLUSTER and VCOMPLEX	5.1 Voltage and power domains on page 39
Clarified description of shared logic clock behavior	5.2.2 Low-power state behavior considerations on page 44
Added information about the *Maximum Power Mitigation Mechanism* (MPMM)	5.3 Power control on page 44
Clarified description of Warm reset	5.4 Core power modes on page 46
Added general TLB information and L1 *TRace Buffer Extension* (TRBE) TLB entry to table	6.1 Memory Management Unit components on page 54

Change	Location
Added additional information after the table	
Added new section	8.2 Write streaming mode on page 66
Clarified introduction text in section	10 Direct access to internal memory on page 76
Added encoding column to table	
Clarified descriptions of how to read RAMs throughout chapter	
Clarified introduction text in section	16.2.1 Core interfaces on page 93
Clarified description of Warm reset	16.2.2 Effects of resets on Debug registers on page 94
Added new section	16.2.3 External access permissions to Debug registers on page 94
Added new section	16.6 CoreSight component identification on page 96
Renamed ROM table registers	16.7 ROM table register summary on page 97
Modified description of performance monitors events `0x00D0` and `0x00D1`. Removed events `0x00D2` and `0x00D3`.	17.1.3 implementation defined performance monitors events on page 110
Modified description of PRESENT bit in the table	C.7.5 ROMENTRY4, Class 0x9 ROM Table Entries on page 553
	C.7.6 ROMENTRY5, Class 0x9 ROM Table Entries on page 554
	C.7.7 ROMENTRY6, Class 0x9 ROM Table Entries on page 556

Table D-5: Differences between issue 0001-10 and 0002-11

Change	Location
First Confidential early access release for r0p2	-
Added new section	2.3 DSU-110 dependent features on page 25
Added new section	5.3.1 Maximum Power Mitigation Mechanism on page 45
Added tables about PCSM power states	5.5 Complex power modes on page 50
Clarified description of cache line allocation into L2 cache for transient memory and for non-temporal loads	8.3 Memory system implementation on page 67
Clarified description of allocation of writes into L2 cache when transient bit is set	9.2 Support for memory types on page 73

Table D-6: Differences between issue 0002-11 and 0003-13

Change	Location
First early access release for r0p3	-
Clarified note about cache indices	2.2 Cortex-A510 core implementation options on page 24
Added missing field [30:4] for table relating to IMP_CDBGDR0_EL3 bit descriptions after a SYS IMP_CDBGL2TR1 operation	B.3.1 IMP_CDBGDR0_EL3, Cache Debug Data Register 0 on page 187
Minor edits throughout section	B.5 Identification register summary on page 214

Table D-7: Differences between issue 0003-13 and 0003-16

Change	Location
Second early access release for r0p3	-

Change	Location
Clarified the meaning of **n** in **COMPLEXCLK<n>**	4 Clocks and resets on page 38
Removed **EVENTI** list item, as this is actually included in the list, as part of architecturally defined WFI or WFE wakeup events.	5.2.1 Wait for Interrupt and Wait for Event on page 43
Clarified description of Debug recovery mode.	5.4.6 Debug recovery mode on page 49
Clarified descriptions of TLB hits	6.1 Memory Management Unit components on page 54
Minor clarifications and corrections	6.5 Hardware management of the Access flag and dirty state on page 58
Added a note about cache maintenance operations	7.1 L1 instruction cache behavior on page 62
	8.1 L1 data cache behavior on page 65
Clarified the location of the link register value in the description of return stack	7.3 Program flow prediction on page 63
Clarified the Preload instructions description	8.5 Data prefetching on page 70
Clarified description of nodes	11 RAS extension support on page 79
Clarified description of the debug system	16 Debug on page 91
Added information about watchpoints	16.2.4 Breakpoints and watchpoints on page 95
Modified some table values for r0p3	16.6 CoreSight component identification on page 96
Clarified note about event export to the *Embedded Trace Macrocell* (ETM)	17.1.1 Architectural performance monitors events on page 99
Modified description of TRCEXTOUT events	
Clarified the Arm recommendations for using the event numbers	17.1.2 Arm recommended implementation defined performance monitors events on page 107
Modified previously incorrect event name to be STALL_BACKEND_ILOCK_VPU. Clarified description of STALL_BACKEND_VPU_HAZARD.	17.1.3 implementation defined performance monitors events on page 110
Modified MPMM event names and descriptions	20.3 Activity monitors events on page 126